Hy pershift Execution Strategies
Demos (regular Hypershift creation):
Part 1: https://ibm.box.com/s/6x98v5av88afe7uw4hqkaxws3914nu2y
Part 2: https://ibm.box.com/s/q7t1r16lnoym4t8msa4ysg2pbfp75b8f

IBM Cloud VPC Gen 2 Specific RedHat CoreOS demos:
Image Import: https://ibm.box.com/s/8su94vcophshmrgn4cqgi5lb1toklsza
Machine Boot Phase 1: https://ibm.box.com/s/4xm2j8xl1mmxviof7fgj5ffkpdxt9voh
MachineBoot Phase 2: https://ibm.box.com/s/jg05dd8sltm617qi4m2tq2mc6bu8n446

Other things

Completed:
Priority class enhancements: https://github.com/openshift/hypershift/issues/410
Determine how we will do different payloads for different providers for RHCOS

In Review:
Assigned:

Hypershift PR integration with E2E deployment of IBM Cloud clusters/RHCOS and e2e tests: https://github.com/openshift/hypershift/issues/652

Not Started:

Component Rollout optimizations: https://github.com/openshift/hypershift/issues/1156

Fix livliness/readiness probe gaps: https://github.com/openshift/hypershift/issues/408

Automatically template armada-hypershift-repo to have appropriate replicas and affinity rules

Hypershift control plane metrics system https://github.com/openshift/hypershift/issues/261

Fix router pod in location (cert issue: minor): https://github.ibm.com/alchemy-containers/armada-frontdoor/issues/6198

Support exposing kube apiserver through router (prototype: not sure if feasible)

Set minReadySeconds on HA deployments

https://github.com/openshift/hypershift/issues/1029

Auditing Profile reconciliation between ROKS/Hypershift:
https://github.com/openshift/hypershift/issues/1040

Priority Class enhancement as node count changes
https://github.com/openshift/hypershift/issues/1041

Deployment reconciliation fails when labels change (roks toolkit):
https://github.com/openshift/hypershift/issues/1042

Operations: Updating nodePool kills all in flight provision operations (No overlapping time with old and new token in place):
https://github.com/openshift/hypershift/issues/1008

FedRAMP/Replatform Items

FIPS of all components: https://github.com/openshift/hypershift/issues/271
Ensure barges are setup

Hypershift for IBM Cloud (VPC Gen 2)

IBM Cloud Cluster API Provisioning system support(IBM only):
https://github.com/openshift/hypershift/issues/231

Test E2E bootstrapping of RHCOS nodes in IBM Cloud:
https://github.com/openshift/hypershift/issues/398

Implementation details for instance group scale request when assigning new nodes in RHCOS

Add hypershift worker-pools for all tugboats

Update hypershift/kubx-deployer to appropriately migrate ROKS toolkit clusters to hypershift

- Remove openvpn operator
- Remove ingress operator in cluster
- Remove in cluster catalog pods and olm operator
- Add toggle for sending through hypershift on version release

========== Initial Sat GA Target ==========

Satellite Hypershift GA

Completed:

Gating logic to not allow users to assign RHCOS workers when a cluster is not hypershift based (Ignition Port check)
https://github.ibm.com/alchemy-containers/armada-ironsides/issues/4146
Provision new hypershift location tugboats (prod):
https://github.ibm.com/alchemy-containers/satellite-planning/issues/1635

In Progress:
Pass region release smoke tests

Satellite GA Enhancements (Operational + GA critical gaps)

Completed:
Audit webhook: https://github.com/openshift/hypershift/issues/232
Rotation of ignition token: https://github.com/openshift/hypershift/issues/364
KMS Support: https://github.com/openshift/hypershift/issues/233
Concept doc: https://github.com/openshift/hypershift/issues/394 **(in drive)**
Portieris support: https://github.com/openshift/hypershift/issues/234
Allow dynamic configuration of resource requests/limits: https://github.com/openshift/hypershift/issues/409
Cleanup stale old nodePool ignition tokens/secrets: https://github.com/openshift/hypershift/issues/411
Monitoring of rhcos update process: https://github.ibm.com/alchemy-containers/rhcos-vpcgen2-ignitiondata/issues/39
Add limits to applicable components
https://github.com/openshift/hypershift/issues/653
Determine how to turn off unsupported locations:
https://github.ibm.com/alchemy-containers/satellite-planning/issues/1607
Proper MZR/HA review of all components: https://github.com/openshift/hypershift/issues/349
Ignition Server Token Rotation Problem: Rotation kills all in flight provision operations (No overlapping time with old and new token in place)
https://github.com/openshift/hypershift/issues/1008
GA Security: All secrets should be mounted to not have global read **(Requires Bom update 4.8/4.9)**:
https://github.com/openshift/hypershift/issues/1031
Architecture MVP: Proper versioning strategy for cluster-api, machine-approver, ignition-server, and cluster-autoscaler
https://github.com/openshift/hypershift/issues/1078
Zonal failure test pass: https://github.com/openshift/hypershift/issues/350

In Review:

Prototyped (Initial redhat review):

Prototyped (No RedHat Review):

Assigned:

GA Operations: Persisting full ignition payload cache so restart of ignition server doesn't cause mass pod scheduling at scale:
https://github.com/openshift/hypershift/issues/1021

Not Started:

Satellite Prod Beta Enhancements:

Completed

Remove kube-admin login
https://github.com/openshift/hypershift/issues/718
Monitoring review of all new hypershift components and associated alerts (ignition server, etc)
Monitoring on ignition server:
Monitoring on hypershift operator:
Monitoring on ingress-operator
Monitoring on control-plane-operator & hosted-cluster-config operator
Monitoring/cleanup on any mcs pods
Monitoring on max nodepools
https://github.ibm.com/alchemy-containers/armada-ops-alert-conf/pull/new/update-ignition-rules
https://github.ibm.com/alchemy-containers/satellite-planning/issues/1598
https://github.ibm.com/alchemy-containers/satellite-planning/issues/1599
https://github.ibm.com/alchemy-containers/satellite-planning/issues/1600
Security: Reduce scope of CPO roles
https://github.com/openshift/hypershift/issues/1026
GA Operations: Ensure ability to pause reconciliation **(Requires BOM Update 4.8/4.9 only)**
https://github.com/openshift/hypershift/issues/1046
GA Operations: Extend resource preservation support to more control plane components **(Requires BOM Update 4.8/4.9 only)**
https://github.com/openshift/hypershift/issues/1067
GA Operations: Pass max scaling test: https://github.com/openshift/hypershift/issues/306
GA Operations: Pass max worker pool test:
https://github.com/openshift/hypershift/issues/810

In Process

Not Started

Ensure BOM (4.9) promoted to prod and passes validation

Restrict ordering of ROKS cruiser clusters to 4.9
https://github.ibm.com/alchemy-containers/armada-ironsides/issues/4049

Stage Satellite CoreOS MVP Capabilities

Completed

Reflect OS data on host so it can be assigned appropriately
https://github.ibm.com/alchemy-containers/satellite-planning/issues/1585

- Reflect OS label in host labels
- API when assigning needs to look at workerpool OS and ensure it matches machine OS (no os == RHEL 7 as default)

Necessary IBM components deployed to environments
Rhcos-management-cluster-config
Rhcos-incluster-config
https://github.ibm.com/alchemy-containers/rhcos-vpcgen2-ignitiondata
Ensure bootstrap labels getting applied appropriately (OS label)
https://github.ibm.com/alchemy-containers/satellite-planning/issues/1585
On Prem RHCOS boot order review/design acceptance :
https://drive.google.com/file/d/1oeMwAYZJmXNV1-aAo21EQ78exRbVdT0P/view?usp=sharing
https://github.ibm.com/alchemy-containers/satellite-planning/issues/1595
RHCOS In place upgrade system:
https://github.com/openshift/hypershift/issues/530
MVP: Control-plane-operator does not properly reconcile when a component changes environment variables (upgrade blocker)
https://github.com/openshift/hypershift/issues/945

In Progress

API templating attach script
RHCOS support for templating attach script

E2E tests on provisioning and removal paths.
https://github.ibm.com/alchemy-containers/satellite-planning/issues/1608

Not Started

API/Armada-cluster logic to manage lifecycle of RHCOS workerpool/nodepool that matches on prem structure

https://github.ibm.com/alchemy-containers/satellite-planning/issues/1596

API/Armada-cluster logic to ensure worker-pools initialized for both RHCOS and RHEL 7 for hypershift locations + hypershift downstream clusters
WorkerPool Operating systems to RHCOS for hypershift clusters

API templating logic
RHCOS support for templating assign script

Stage Satellite Hypershift MVP Capabilities (No CoreOS)

Completed:

Ensure Secure additions to master namespace are there for Satellite tugboats (central named cert and iam id):
Update OS_ID label for CoreOS machines
https://github.ibm.com/alchemy-containers/satellite-planning/issues/1587
Determine how to toggle platform config (Satellite + AWS for example)
Add toggle logic for armada-deploy to deploy openshift location clusters to dedicated tugboats:
https://github.ibm.com/alchemy-containers/armada-deploy/issues/7146
Armada-API support to allow beta users to order Openshift based location. By default order iks based location:
https://github.ibm.com/alchemy-containers/armada-api/issues/5772
Ensure Konnectivity Port is sent to satellite users in same way as Openvpn port (api level)

- (Exposed at cluster.`KonnectivityServerPort`)
- `Just need API support`

https://github.ibm.com/alchemy-containers/armada-ironsides/issues/3819

Ensure IgnitionServerPort is sent to satellite users in same way (api level)
https://github.ibm.com/alchemy-containers/armada-ironsides/issues/3820
Fix ingress policies:
https://github.com/openshift/hypershift/issues/925
RedHat Konnectivity build and push to all IBM registries (with signature IBM Only). Support from network-squad:
https://github.ibm.com/alchemy-containers/armada-konnectivity-community-build/issues/24
https://github.ibm.com/alchemy-containers/armada-konnectivity-community-build/issues/25
Test core functionality:
https://github.com/openshift/hypershift/issues/397
Patch pipeline agreement for all components mainly
https://github.com/openshift/hypershift/issues/414
Konnectivity MVP performance fixes (socket leak causing excessive memory use and deadlocks):
https://github.ibm.com/alchemy-containers/armada-network/issues/5866
https://github.ibm.com/alchemy-containers/armada-network/issues/5329
https://github.com/kubernetes-sigs/apiserver-network-proxy/issues/261
https://github.com/kubernetes-sigs/apiserver-network-proxy/issues/255
Provision new hypershift location tugboats (staging):
https://github.ibm.com/alchemy-conductors/team/issues/13798
Fix gap in ingress operator canary health checking:
https://github.com/openshift/hypershift/issues/1130
Security: Eliminate dynamic non IBM Cloud images in the controlplane **(Requires BOM Update 4.9 only):**
https://github.com/openshift/hypershift/issues/1075

In Review:

Assigned:

Hypershift feature branch merged and built into a BOM prior to GA **(Requires BOM Update 4.8/4.9 only)**
https://github.ibm.com/alchemy-containers/armada-update/issues/2909

Not Started:

========== ARCHIVE LINE ==========

Ability to run Hypershift in IKS based locations

https://github.ibm.com/alchemy-containers/satellite-planning/issues/1602

Auto non root security contexts

Internal tracker:

Inplace bootstrapper support for CoreOS
https://github.ibm.com/alchemy-containers/satellite-planning/issues/1586

RHCOS Dev Preview

Completed:
Ignition payload generation critical hardening: https://github.com/openshift/hypershift/issues/343
RHCOS ignition server authentication: https://github.com/openshift/hypershift/issues/218
IBM RHCOS update job system/custom config rollout:
- need to know what all we will restart/need to automate in this process:
https://github.com/openshift/hypershift/issues/303
https://github.com/openshift/hypershift/issues/225
Dynamically find proper haproxy router image:
https://github.ibm.com/alchemy-containers/rhcos-vpcgen2-ignitiondata/issues/30
RHCOS image import system (IBM only):
https://github.com/openshift/hypershift/issues/396

Dynamically update core RHCOS config on changes:
https://github.com/openshift/hypershift/issues/386
TLS for ignition payload download to ignition server:
https://github.com/openshift/hypershift/issues/354
RHCOS HA multi zone ignition server:
https://github.com/openshift/hypershift/issues/406
Performance tuning for RHCOS:
https://github.ibm.com/alchemy-containers/armada-bootstrap-squad/issues/876
secure kubelet serving/client cert approving system (machine approver integration):
https://github.com/openshift/hypershift/issues/279

In Review:

Prototyped (Initial redhat review):

Prototyped (No RedHat Review):

Assigned:

Not Started:

Dev Preview

Completed:
Audit webhook support
Control over Konnecitivity image or build in release image: https://github.com/openshift/hypershift/issues/348
Remove need for log connections in operations: https://github.com/openshift/hypershift/issues/266
Multi-zone deployment support: https://github.com/openshift/hypershift/issues/369
Pod affinity based off clusterid: https://github.com/openshift/hypershift/issues/370
Ability to restart master components with “restart annotation” (follow assign with Cesar): https://github.com/openshift/hypershift/issues/376
Liveness/Readiness Probes: https://github.com/openshift/hypershift/issues/240
Priority class specification: https://github.com/openshift/hypershift/issues/238
Resource request and limit specificates for accurate scheduling: https://github.com/openshift/hypershift/issues/239
MVP: determine strategy for hard coded cluster-api/autoscaler: https://github.com/openshift/hypershift/issues/412
Tolerations on master components: https://github.com/openshift/hypershift/issues/371
Openshift APIServer Konnectivity proxy integration for webhooks and on prem registries: https://github.com/openshift/hypershift/issues/341
Disable CSR auto approver:
Determine In Cluster Catalog content strategy https://github.com/openshift/hypershift/issues/351

In Review:

Prototyped (Initial redhat review):

Prototyped (No RedHat Review)

Assigned:

Not Started

Prestage Dev Preview Work Tracker Satellite

Completed
External etcd: https://github.com/openshift/hypershift/issues/219
Konnectivity Integration Phase 1: https://github.com/openshift/hypershift/issues/222
Cert PKI adoption: https://github.com/openshift/hypershift/issues/226
DNSMasq removal or build process
https://github.com/openshift/hypershift/issues/315
Network Type: https://github.com/openshift/hypershift/issues/223
Named Cert support: https://github.com/openshift/hypershift/issues/230
Oauth Identity Provider: https://github.com/openshift/hypershift/issues/224
Pod CIDR/Service CIDR/Advertised Address and secure port:
https://github.com/openshift/hypershift/issues/228
Fix new-app https://github.com/openshift/hypershift/issues/353

In Review:

Initial Openshift-master PR review:
https://github.ibm.com/alchemy-containers/armada-openshift-master/pull/928/files

Prototyped (Initial redhat review):

Prototyped (No RedHat Review):

Not Started:

RedHat + IBM High level Work Streams:

Authentication for Machine Config Server:

Current

Machine Config server has no authentication and anyone that has network access to the server can download sensitive cluster payload

Solution:

Will add a haproxy sidecar that verifies a token url parameter that corresponds to the node-bootstrapper-token in the cluster. Machines in their boot user-data will specify this token and be able to fetch the payload. Controllers will be in place for when the token is rotated to update the user-data payload and haproxy config to reference new token automatically.

Example:

```

{"ignition":{"config":{"merge":[{"source":"https://ignition-provider-master-clusterid2.bts-prestg-bare-840615-15b0bf3714b2d9d344c3b8647edb7860-0000.us-south.stg.containers.appdomain.cloud/config/master?token=ZXlKaGJHY2lPaUpTVXpJMU5pSXNJbXRwWkNJNklsVkpVekkwUVVOSWRqRkpaMFJyWTNCamIwTmxXRFJJUlhZMFRrNXROa05HUWpkQmIybzVXa05tVVdjaWZRLmV5SnBjM01pT2lKcmRXSmxjbTVsZEdWekwzTmxjblpwWTJWaFkyTnZkVzUwSWl3aWEzVmlaWEp1WlhSbGN5NXBieTl6WlhKMmFXTmxZV05qYjNWdWRDOXVZVzFsYzNCaFkyVWlPaUp2Y0dWdWMyaHBablF0YVc1bWNtRWlMQ0pyZFdKbGNtNWxkR1Z6TG1sdkwzTmxjblpwWTJWaFkyTnZkVzUwTDNObFkzSmxkQzV1WVcxbElqb2libTlrWlMxaWIyOTBjM1J5WVhCd1pYSXRkRzlyWlc0dFp6azJjbmNpTENKcmRXSmxjbTVsZEdWekxtbHZMMM05sY25acFkyVmhZMk52ZFc1MEwzTmxjblpwWTJVdFlXTmpiM1Z1ZEM1dVlXMWxJam9pYm05a1pTMWliMjkwYzNSeVlYQndaWElpTENKcmRXSmxjbTVsZEdWekxtbHZMM05sY25acFkyVmhZMk52ZFc1MEwzTmxjblpwWTJVdFlXTmpiM1Z1ZEM1MWFXUWlPaUkwWm1SbVl6WTVNeTFtTVRnMExUUTBNRGt0WVRVeU15MDJaVFpqTm1NNE9EWXdaRGtpTENKemRXSWlPaUp6ZVhOMFpXMDZjMlZ5ZG1salpXRmpZMjkxYm5RNmIzQmxibk5vYVdaMExXbHVabkpoT201dlpHVXRZbTl2ZEhOMGNtRndjR1Z5SW4wLkdDNXAwR1BFMV8xZW5fUDA0YmF3M3Z6SUR2SzZsaVFLOXlMZjU0WmxDN1FfclV2WU9TcGxLUEVGOWlEM3JKSlNiVU5Ia0liQnFZY0tDeE1xSWZBZEluUDgwOElWalQ3Tm1iQkM0QUFnS2tJMFUxQW4tOTZVUmhDQ2UyNWRYdG55c0xlc0UyRzBrRXRFeGFuNU5HZFdDZzNLOHctbUNUN05RSXp2dUZGZkh0NFFkbW9iOVM3alF1cDRsVzFvR0p0UlRsMWFoWmtZRGdjeUhkSHA0cmhGT2k0NWt0OW1KNEpreDZYR1dkdmhmZmxlUzRoaThKUnU3dGt4aV9PS2tGMFRSTmExSWtIQk1WcmpWbVM4dGNmeXNGWmNrdzZXaHNNcjVqNzBoUXdOU2RwQ3VrbkxYVndnVUhfRDFBWFl6eXdHXzVoSzZEWXRtcHlmZXBkZFFuVy1LUQ%3D%3D","verification":{}}]},"security": { "tls": { "certificateAuthorities": [ { "source": "data:text/plain;charset=utf-8;base64,LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURFRENDQWZpZ0F3SUJBZ0lJSlhPV3BCVnpaZDh3RFFZSktvWklodmNOQVFFTEJRQXdKakVTTUJBR0ExVUUKQ3hNSmIzQmxibk5vYVdaME1SQXdEZ1lEVlFRREV3ZHliMjkwTFdOaE1CNFhEVEl4TURNd09USXpOVGt6TkZvWApEVEk1TURNd056SXpOVGt6TkZvd0pqRVNNQkFHQTFVRUN4TUpiM0JsYm5Ob2FXWjBNUkF3RGdZRFZQUURFd2R5CmIyOTBMV05oTUlJQklqQU5CZ2txaGtpRzl3MEJBUUVG
```

QUFPQ0FROEFNSUlCQ2dLQ0FRRUF0QktUY3F5cjBnY08KK2h6UWduVVdJT2czT
WxYTXY2SUNoNVJ5TmVRZVhCeEtzQUF2T1VYaDhFa0hTbTFLbUJpemJVTVFxc
DA2Zm9XeApUVlZWd2pPMkd0eGN6UHd3dmduMlR0SFRielNHTkV5a05ESlhOR1
MrYVY3cFlBVkl4TFN6bmhpZUtYRDdaa1BrCkVtODFZck5rTG9vRkhXZXJZbFYwSn
NWa1Q3SjdQYXk0R1d4aDduTDE3aEpUdnRCYkRvc0lMbTJmT01ZelhEeCsKamcx
Q0VlOWUrYkFWUHhoZjFvSTJiYnNLRDBST2tCc2N3T0p5VGpYWWk4RUpYd3hDe
GpqeWZQb0hCc0h2TnpRUQpjODNHTXRxVkEyR1VBUmdITzk4V0VmYUpzYlNWT
ys4SlZ5cWs3c2FhUkZYeFcyYVRSUndOOGhQMDZkVGNZTld2CmVFN0tEcDVXMl
FJREFRQUJvMEl3UURBT0JnTlZIUThCQWY4RUJBTUNBcVF3RHdZRFZSMFRBU
UgvQkFVd0F3RUIKL3pBZEJnTlZIUTRFRmdRVVBGYmdkWXVOdlhXS1k5cVZWVU
o5cnF5Q1E4WXdEUVlKS29aSWh2Y05BUUVMQlFBRApnZ0VCQUR2QjJhSVVTZn
hnRXJPQXBsYXdBb3A4WXNFazVtVnpLYmNKSWE4ODU4bXNLelNYckhPUnBkY1
FrR0haCmZQOUc0T3dyVVAzbXdPbFNqYmo1cWZ4c04vV1pHNXZpMTh1aVVLQU
NZZWZpanlVRWhqT0dBZll2K1ArWlVQWXgKVG1IUms1WGI4UjU1Zm9CMXVIb0pV
S0ttR0hhLzRZOE9yWWVSd2VyYkFsbnF0TVZ6NFh3bzZZcTVHSnh3M1BpdAova2Z
VWmoxQzVjeVdsZVpGKzQ5NDNOSklrMy8xd1dsWllzMWx6Rk82MUdpWWpDdEN
VN2VtVys4U1crMkRmWmdiCmQ1NjZSTC9xR0tGRDZFa2F6bXA0R1AzMk42OUVB
Zks2aXBRNWtERGs3MStBdjd0UzVhMFNDUk8vZnJyU3ljU20KaDRabWtRUytqZzIw
d2x4a04ydnkrVU1peXJzPQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCi0tLS0tQ
kVHSU4gQ0VSVElGSUNBVEUtLS0tLQpNSUlESGpDQ0FnYWdBd0lCQWdJSWVM
OVNuV05QNkpnd0RRWUpLb1pJaHZjTkFRUxCUUF3TFRFU01CQUdBMVVFCkN
4TUpiM0JsYm5Ob2FXWjBNUmN3RlFZRFZRUURFdzVqYkhWemRHVnlMWE5wWjI
1bGNqQWVGdzB5TVRBek1Ea3kKTXpVNU16UmFGdzB6TVRBek1EY3lNelU1TXp
SYU1DMHhFakFRQmdOVkJBc1RDVzl3Wlc1emFHbG1kREVYTUJVRwpBMVVFQX
hNT1kyeDFjM1JsY2kxemFXZHVaWEl3Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVV
BQTRJQkR3QXdnZ0VLCkFvSUJBUURlMW5qUFExL2JtM1U1N0g3dEltNWw3VTFN
S3ZPYUZqeDF4dDc5aWdiQW4veFRMMGNQSFpWczRWbnIKOVRTVklHWnFtdXp
DREpURXcrWjhRUWlUTXpFRnkzY0F6Tmp4c1Y4ZFlZeW44QTJxZ3NCNks4WkluT
VM2S1kxRwpjY1RiYWV2RzVJVHVEN01CaHllUGpOSFRvcjVlbHhKczlibGtxU1JMV
GxwWFFOQ2x1ZU5LeHZXUDk1M1dtVDQ2ClYybTNwVlEwUk80ZGliTy9uMEJlUEFr
YmtzUlk3ZzVwdCtrNFVrSWpOSzhnYlprelBTSUxJM0ZRL2thU3dlLysKeW8xZjhYW
WJOZEI2WXVqcHJvV2k0ZHJXVHZnQVd0OEhCcXJTZ0lldUJCMzhNeXVaZnUrdGh
2eGhRNVhYd1pxeAo5SzNWRGYzamFaN0JPYXlBUHE4YlRvY21RRitsQWdNQkFB
R2pRakJBTUE0R0ExVWREd0VCL3dRRUF3SUNwREFQCkJnTlZIUk1CQWY4RUJ
UQURBUUgvTUIwR0ExVWREZ1FXQkJTSkRTOHZtanVBY3VSaldPenA1Uk1udTJT
RENUQU4KQmdrcWhraUc5dzBCQVFzRkFBT0NBUUVBSUIyUW44eXdyOSt6MHE
2MG5haVBTVFdXL3dQU2NKY2FETUhGcDJZTApoR0FCelIzUVpmMUVtTHhrQVd6
M2ROQ3U2M1U5djdSNUlwdTNTUWJ1b0pqWm1vWHZxajNKTnZSem5BTEhaN1Vt
CkZoZHB0UjdBUXRHQkhsY01LbU1UQUFMdFFCY2w1azRqSE1vaHp0cGpyRFpxZ
jZmKy9XcE9DOE9jdHk3TjZmTGYKYktKUHMzNXJ3bkdoYVBIRlpxQnFpSUxaVHUx
aXBTNlpMaHN6Z05WTEltNC9GaHFxeW53SXBPM2RVb0t0WE5hNQpidGUvaFJZQ
U04c3pOVHdJZEltOVhrRDZxa0RpZnIvRnZJOXc4bDlPeTJFS3VoK0Q4RmFxMVdk
YnZkQzUzQ1RyCktYVll5by9ZV0VVUnVRNjN5WU5qeTc5K0JycG9LODA0WjNXaHR
Nbk5oTXI3S1E9PQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==",
"verification":{} }] }
},"timeouts":{},"version":"3.1.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}

```

**Control Plane Publishing strategy:**

Current:  they rely on provisioning individual routers per cluster and applying routes for oauth and ignition servers. Routers are exposed with load balancers along with the kube-apiserver service and oauth service

MVP:

Ability to publish all services using node ports

**Exit criteria:**
All external services exposed as a node port which include:

```
kube-apiserver   NodePort   172.21.217.92 <none>
2040:31320/TCP   39h
machine-config-server   NodePort   172.21.95.205   <none>
80:31249/TCP   39h
konnectivity   NodePort   172.21.55.150 <none>
1194:31250/TCP   39h
oauth-openshift   NodePort   172.21.162.205   <none>
443:31251/TCP   39h
```

Long term (ignore for this year):

Utilize SNI https://en.wikipedia.org/wiki/Server_Name_Indication and unique domain names per cluster per exposed service to be able to route everything through only port 443 using ISTIO

Example

konnectivity-clusterid1.SUBDOMAIN -> istio/router -> backend service TLS passthough
apiserver-clusterid1.SUBDOMAIN -> istio/router -> backend service TLS passthough
oauth-clusterid1.SUBDOMAIN -> istio/router -> backend service TLS passthough
```

and deploy config that will tell the router component to send TLS to these endpoints.

Benefit:

Customers will only need to expose 443 to the set domain name
All ingress traffic comes through single point of entry that can enable advanced audit log use cases

"Adopt" existing clusters/Asset payload control ("bring your own ETCD")

Current:

In automation upstream etcd operator is deployed and a single replica etcd cluster is deployed as well

Requirement:

Need to be able to seamlessly "adopt" all the existing ROKS clusters we have in production to be under hypershift control

One key difference: etcd clusters are named differently in ROKS and automation deploys etcd-operator/the etcd cluster. All backup/restore/operations associated with etcd depend on this system.

Example: named etcd-CLUSTERID. All services correspond to that name as well
kubectl get etcdcluster -n master-bvnqi2710tsfq5mqtbrg
NAME AGE
etcd-bvnqi2710tsfq5mqtbrg 68d

We will also create the downstream PKI secrets that integrate into our CA rotation system.

Worker management through Cluster API (Specifically targeting specific workers behind pool for upgrade/removing)

Want to talk about individual node management. Specifically how lifecycling of individual nodes in a node pool works out.

This is around some smaller scale cases where people do targeted things on specific workers and if a group was replaced at once it could cause problems

One example: 2 edge nodes in a cluster that holds ingress/router components

User requests upgrade and it's all automated but causes downtime
Can a user specifically target upgrades of one node at a time?
What do we want to allow for a lifecycle at the individual worker node level.

Internal IBM (no Redhat involvement)

Ignition payload ("First boot" payload) enhancements to support IBM Cloud

Requirement:

We need to adjust the ignition payload to support our IBM Cloud environment. One of the obvious gaps is we need to get the nodes to boot with IPs in the IBM Cloud environment currently.

Cluster-API machine controller for IBM Cloud VPC Gen 2

Requirement:

A machine controller needs to be designed for the cluster-api project that supports IBM Cloud environments.
Example for AWS that works E2E:
https://github.com/kubernetes-sigs/cluster-api-provider-aws
Research prototype for IBM:
https://github.com/kubernetes-sigs/cluster-api-provider-ibmcloud

Import system for Red Hat CoreOS machines for IBM Cloud VPC Gen 2

Requirement:

We need to add support for pulling coreOS images https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.7/4.7.0/rhcos-4.7.0-x86_64-ibmcloud.x86_64.qcow2.gz , resizing them to match the boot disk size of gen 2 environments, and then publishing them in all supported Gen 2 accounts
Resizing command: qemu-img resize PATH_TO_QCOW2_IMAGE 100G
Script that does this for regular IKS/Openshift today:
https://github.ibm.com/alchemy-containers/armada-softlayer-image/blob/1b9a532ab6c11cadfdb1a7218c2ad974ec4b414f/gen2image/import-image-to-cos.sh

Similar thought process just instead of initially doing export_vhd_image_to_cos we will start by pulling the RHCOS image

API/Billing Workflow centered around the NodePool concept outlined in the Hypershift repo:

Requirement:

We need to have APIs for the zone-add, resize, worker replace, etc that are all centered around CRUD operations to the CRDS that will exist in the tugboat.
For example: zone-add will ultimately need to create a NodePool CRD in the tugboat that is running the master components.
For any data we need reported back to etcd: component(s) can be designed that look for changes in these CRDS and send the necessary data back to armada-etcd.
However, whenever possible we should prefer to keep data local to the tugboat in CRD form

Bulleted hypershift work item list

- MVP of HyperShift for ROKS (with RHEL7 hosts)
    - Konnectivity integration (IBM)
        - No VPC UDP load balancers
        - No FIPS package for openvpn
        - IBM team to work with seth/cesar to find a home for this
        - Openshift-apiserver also needs access to konnectivity proxy
        - Single connection can block others, likely needs to be resolved upstream
        - https://github.com/kubernetes-sigs/apiserver-network-proxy/issues/180
        - This be a switchable option
        - Scupper?
        - https://github.com/openshift/hypershift/issues/222
    - Cloud provider metadata support for IBM Cloud VPC Gen 2
        - “If ibmcloud then….”
        - Cesar: potentially use a CR to control lifecycle
        - https://github.com/openshift/hypershift/issues/220
    - Calico SDN/Tigera Operator integration
        - Need to be able to send in calico rather than OCP SDN
        - Derek: Any 3rd party CNI should keep working as a goal of Hypershift. (networktype: none, customer supplies OLM managed CNI)
        - https://github.com/openshift/hypershift/issues/223
    - IBM Cloud IAM integration with Oauth server
        - Plan is to support oauth server idp configs
        - https://github.com/openshift/hypershift/issues/224
    - "Cert/PKI adoption" to ensure existing CA rotation pieces are supported
        - IBM generates and manages its own certs/domains for clusters. Need to feed these into the hypershift-operator
        - https://github.com/openshift/hypershift/issues/226
    - Multi-zone deployment support (3 multi-zone replicas)
        - Ability to have custom affinity policies for control plane pods
        - https://github.com/openshift/hypershift/issues/227
    - Integration with externally managed etcd (etcd client url parameter specifications)

        - Yes
        - https://github.com/openshift/hypershift/issues/219
    - Pod CIDR/Service CIDR/Advertised Address and Secure Port control
        - Yes
        - https://github.com/openshift/hypershift/issues/228
    - ~~CSR Cert signing duration to match CISO timelines (controller manager parameter)~~
        - ~~https://github.com/openshift/hypershift/issues/229~~
    - Named cert support to integrate Digicert issued certs for E2E TLS validation for clients using console/etc
        - Something that OCP needs to expose in general
        - https://github.com/openshift/hypershift/issues/230
- RHCOS enablement for IBM Cloud
    - ignition/Machine Config Server authentication/authorization
        - https://github.com/openshift/hypershift/issues/218
    - Enhancements to allow a single ignition server to serve multiple independent config pools/Ability to specify additional VPC Gen 2 cloud-specific ignition configuration
        - https://github.com/openshift/hypershift/issues/225
    - IBM Cloud VPC Gen 2 cluster-api machine controller integration
        - https://github.com/openshift/hypershift/issues/231
- ROKS Feature parity
    - API Server Audit webhook support: https://kubernetes.io/docs/tasks/debug-application-cluster/audit/
        - https://github.com/openshift/hypershift/issues/232
    - KMS support for secret encryption: https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/
        - https://github.com/openshift/hypershift/issues/233
    - Portieris support for verifying image signatures: https://github.com/IBM/portieris
        - https://github.com/openshift/hypershift/issues/234
- OPERATIONAL OPTIMIZATIONS
    - Pod affinity based off clusterid to minimize spread of workload of master components for a given clusterid
        - https://github.com/openshift/hypershift/issues/235
    - Ability to restart master components with "restart annotation" on master refresh operations (no patches applied)
        - https://github.com/openshift/hypershift/issues/236
    - Toleration specification on deployments so taints can be used to isolate workload in management cluster.
        - https://github.com/openshift/hypershift/issues/237
    - Priority class specification so critical deployments aren't evicted
        - https://github.com/openshift/hypershift/issues/238
    - Resource request and limit specifications for accurate scheduling
        - https://github.com/openshift/hypershift/issues/239

        - Liveness and readiness probe specifications on deployments for accurate health checking/monitoring
            - https://github.com/openshift/hypershift/issues/240
        - Upstream IBM alertmanager and prometheus configs
            - https://github.com/openshift/hypershift/issues/241
- Include hypershift images in release image
    - https://github.com/openshift/hypershift/issues/242
- Notable additions to enable support in existing IKS Tugboats
    - Registry image mapping support (Translating images from quay.io to mirrored regional IBM Cloud registries for control plane components)
        - https://github.com/openshift/hypershift/issues/245
    - Security context specification
        - https://github.com/openshift/hypershift/issues/244

Small fix work item list

- Kubelet Kubeconfig should be sending traffic through haproxy pod ip not domain
-

Milestone Tracker for Runtime

Milestone: Go through hypershift workflow in a provisioned Openshift 4 cluster as management cluster and ensure hypershift cluster is deployed successfully (pure upstream)

This is more of a simple familiarization task

Milestone: Go through hypershift workflow in a provisioned IKS cluster as management cluster and ensure cluster is deployed successfully (pure upstream)

This is more of a simple familiarization task

Milestone: Enhance PR tests to test hypershift workflow

Will be small scale until reliance on load balancers removed but plumbing should be put in place.

We can likely use same razee target in the cruisers section just for the test cluster to test that path at PR time
Will then enhance kubx-deployer/armada-openshift-master to apply the HostedCluster CRD properly instead of templating and applying yamls through ibm-roks-toolkit

Ensure PR tests are passing and we have full coverage on this path.

The path from old version to new version that uses CRD will test the adoption path appropriately

Setup testing Openshift 4 clusters (FEDRAMP) as well as IKS clusters for coverage of both paths

VPC Gen 2 pipeline PR account that we can publish test images to over time

Initial worker side testing can be done using AWS account

Milestone: ensure the wrapper components we deploy master side outside base hypershift/ibm-roks-toolkit have a deployment model for IKS tugboats and Openshift tugboats or remove them if possible

We have opportunity for openshift tugboats if applicable to try and move to a different deployment model for some of these addons like through cluster updater if it eases our load

We basically have to ensure whatever we do in our automation functions in both environments (we can control what we enable in an automation suite)

Key impacts I see: PSP bindings for master components (hopefully can turn off), Potentially etcd operator deployment, Openvpn operator deployment, cluster health, cloud provider

Milestone: Install hypershift operator in dev/prestage tugboats

This is a centralized operator so can be installed through razee (won't be through kubx-deployer: kubx-deployer will create CRDs to talk to it)

Milestone: Work with Redhat to get MVP requirements for existing IBM tugboat support: https://github.com/openshift/hypershift/issues/130.

In this specific phase we want to implement the Node Port publishing strategy at a minimum

Machine Config APIServer should have auth enabled for security purposes

No need to focus on adopt: main goal here is trying to get it to performance team for scale testing

Milestone: Test kubx-deployer builds passing in dev and prestage using hypershift operator

Milestone: Manually Create NodePool CRDs and ensure that the provisioning process is working in the environments

We can start with AWS since that cluster-api controller is already there

When manually testing in dev/prestage: we should be simulating the “outlined” api/cluster paths by hand to ensure integration with bootstrap/other components)

There is labeling code at a minimum in bootstrap that will need to be updated but general validation will be important and iteration.

Ideally in May Milestone: Begin performance testing and ensure results are accepted (publish test boms for performance to utilize and install hypershift operator in their tugboats)

May need to get something small to the perf squad that can help them automate creation of node pools.

Execute following scale test 440 clusters *12 node pools per cluster = 5280 node pools

Key focus on load on master and how the single operator approach scales

Iterate on changes to operator.

Focus on outlining scaling parameters and ensuring we are ok on upgrade path in all existing tugboats. For example: at a minimum a new node port is likely being added with machine config server.

Milestone: Ensure IBM Cloud RHCOS image pipeline management in place

Strategy for image imports into our environments over time

Ensure image cleanup strategy also in place (clean old dead images)

How do we bring in what the community offers? Is that every patch update?

Milestone: Nodepool update system (may be nop)

We need decoupling of image updates here from control plane rollouts

Basically how do the nodePool CRDs change over time and get updated with the latest images on each Openshift release? Is this seamlessly handled

How does machine config adjust over time as well?

How do we decouple specific things for faster velocity from BOM rollouts?

I currently believe this will all be seamlessly handled through the operator on upgrade of the control plane release image. The hypershift operator reads that and updates the AMI from it but we should talk to the Red Hat team just to confirm.

Milestone: Install hypershift operator in stage tugboats

Milestone: Ensure that the automated workflow with API/Cluster meets all requirements

Milestone: Work with Red Hat team to add additional support that we deem necessary like KMS, Portieris, etc. Get up to feature parity that we have with ibm-roks-toolkit

Milestone: Hypershift operator installed in stage tugboats and get tests passing

Milestone: Hypershift operator installed into all tugboats

Key IBM Milestones

Milestone: Integration of IBM Cloud cluster API VPC Gen 2 controller in with hypershift

Milestone: API/Cluster workflow implementation

Key next steps from meetings:

Work out individual worker lifecyle flow and reqs (targeted update/replace)

Meeting Notes:
2021/03/18
Control Plane Publishing Policy Follow on:
API server endpoint and VPN cannot be routed through our router with SNI

API Server it was exec/proxy connections specifically failing
VPN Server UDP based but can be solved with konnectivty

Can Ingress config be used to front TLS services instead of Routes (similar to SNI strategy):

Technically possible: but will change network flows that have potential to cause problems

Machine Config Server Auth follow on:
Point made on that clear introspection of traffic point required (basically some smart entity that is used to process all traffic from user cluster to management cluster). This can be something like router or istio ingress gateway. Good for auditing

General idea looks promising using node-bootstrapper-token however needs to be reviewed by stakeholders on RedHat side

Also question around if a node token from a redhat perspective needs to be done on an individual pool basis? Does rotation need to be built into the offering from a Red Hat perspective?

Does the platform rotate tokens? Or does it delegate to external actor?

IBM perspective: we need to have a way to rotate but that can be doced and not automated on a regular time basis. In this strategy it would simply be removing the service account token secret and a new one will be generated. Kubectl delete secret

Import image follow up RHCOS:
Cesar was just curious what are the impacts in the Gen 2 environment on the fact that their is no global image catalog. It means we have to handle pushing the images before machines can utilize them into the account from upstream (and resize to appropriate disk size 100G)

VPC Gen 2 team does allow ordering based off of name of image which we control

So we still could have set asset payload with set names. System just has to be in place either a priori or dynamically to download the image and upload it to VPC Gen 2 account with the proper name (Gen 2 team provides APIs to do that)

Existing cluster support:
Derek and team understand and agree we don't want to force a hard delete/recreate for our existing user base in IKS tugboats

Timeline/execution talks RHCOS:
4.8 is aggressive and potentially not feasible. More end of year is feasible
That's against Fedramp timelines for IBM Cloud.
Basically should stay in sync with Derek and team and if hypershift is not ready have them advise on what intermediate steps to take that doesn't deviate us for end goal

Key automation outside control plane apply (wrapper kubx-deployer):

Removes calico service endpoint policy blocking public traffic if disabled

Fetching of Digicerts for the tugboat cluster to feed into master automation

Satellite: tear down link resources

Creation of master namespace

Checking and creation of cluster roles that bind to psps that masters use

Service creation
- Kube-apiserver
- Openshift-apiserver
- Openshift-oauth-apiserver

Gets PKI data if exists and unarchives

Generates/Regenerates certs

Applies downstream secrets/ CA cert configmaps

Applies KMS config if applicable

Creates oauth cert/client certs, service and ca config map

Creates audit logs config map if enabled

Creates etcd

Renders and applies OC manifests

Applies Open VPN operator

Applies IBM external cloud provider

Applies cluster updater/razee resources

Begin Cruiser cluster paths

Creates IBM-system namespace for load balancers

Creates IBM defined sccs

Installs calico operators

Applies calico configurations

Applies keepalived watcher

Applies icr pull secret and then openshift pull secrets (ibm-rbac)

Removes some rbac that is security problem in ibm cloud

Adds storage roles/secrets

Configures registry by creating pvs/ Image registry config, etc

Creates some CSI volume snapshot classes/CRDs

Applies cluster health master side

Applies calico master network isolation policy

Applies in cluster calico policies (block kubelet, etc)
Applies customizations (ConsoleLinks, etc) for Openshift Console for IBM

Applies RBAC operator in control plane

Applies priority fairness flow control objects

Creates razee roles/rolebindings/configs (satellite)

FINAL PLAY - mix of both

Calico service endpoint apply to block public traffic in private only clusters

Stores cluster artifacts that are written to etcd

Backs up etcd data

Cleans rbac for cluster health and cluster updater (kubx-masters namespace)

Generates cluster updater gpg key

Pushes dynamic addons generated to cluster store

Scales cluster updater up

Key configuration to verify in Hypershift:

IKS tugboat based

Cluster-version-operator:
Restart date annotation support (restarting components when requested by customer)

ClusterID label support (pod affinity)
Toleration specification (tugboat workers all tainted)
Priority class specification
Affinity rules
Security context specification (run as non root)
CPU/Mem requests and limits

Control-plane-operator:
ClusterID label support (pod affinity)
Restart date annotation support (restarting components when requested by customer)
Toleration specification (tugboat workers all tainted)
Priority class specification
Affinity rules
Security context specification (run as non root)
CPU/Mem requests and limits

Kube-Apiserver:
Config.yaml:

Pod Cidr
Service CIDR
Advertised Address
Audit webhook support
Cloud provider specification
Etcd client name
Feature gates
API Secure port
Named cert specification for urls

Deployment.yaml:
Replica count
ClusterID label
Restart annotation support
Toleration specification
Affinity rules
Priority class support
Security context
APIServer verbosity
Liveness probes/Readiness probes
CPU/Mem requests limits
KMS integration
Portieris enablement
Audit webhook support

Feature gates

Kube-Controller-Manager:
Config.yaml:

PodCIDR
Service CIDR
Feature gates
Cluster signing duration

Deployment.yaml
Replicas
ClusterID label
Restart annotation support
Toleration specification
Affinity rules
Master priority class
Security context
Feature gates
CPU/Mem requests
Liveness/readiness probes

Kube-Scheduler:
Deployment.yaml:
Replicas
ClusterID label
Restart annotation support
Toleration specification
Affinity rules
Master priority class
Security context
Feature gates
CPU/Mem requests
Liveness/readiness probes

Oauth-Openshift:
Oauth-browser-client.yaml:
Route to oauth server in management cluster

Oauth-challenging-client.yaml:

Route to oauth server in management cluster

Oauth-server-config.yaml
IBM Cloud IAM identity provider specification
Named cert specification

Oauth-server-deployment.yaml
Replicas
ClusterID label
Restart annotation support
Toleration specification
Affinity rules
Master priority class
Security context
CPU/Mem requests
Liveness/readiness probes

Openshift-Apiserver:
Config.yaml:
Ingress subdomain
Etcd client name

Openshift-apiserver-deployment.yaml
Replicas
ClusterID label
Restart annotation support
Toleration specification
Affinity rules
Master priority class
Security context
CPU/Mem requests
Liveness/readiness probes

Openshift-Controller-Manager:
Cluster-policy-controller-deployment.yaml/Openshift-controller-manager-deployment.yaml:
Replicas
ClusterID label
Restart annotation support
Toleration specification
Affinity rules
Master priority class
Security context
CPU/Mem requests

Liveness/readiness probes

Config.yaml:

Roks-metrics:
Need to read more on the setup to see

“Master Communication” architecture

This section aims to describe the 3 common patterns of network communication that exist in a hypershift cluster with the “exposed services” in the management cluster

Kube-APIServer:
External User client (jenkins, on laptop, etc): client -> *external client network* -> nodeport/load blancers (through url) -> kube-apiserver

User cluster components (calico, coredns, kube-proxy, kubelet, etc): component -> haproxy (local on node) -> *client network* -> backend urls -> kube-apiserver

Management cluster components (control-plane operator, etc): component -> local Kube service entry -> kube-apiserver

Ignition Server:
Worker node: node agent -> *client network* -> nodeport/load balancers/routes (through url. If router also goes through router pod) -> Ignition server

Konnectivity Server:
In cluster Konnectivity agent: agent -> *client network* **->** nodeport/load balancers (through url) -> Konnectivity server

Kube-apiserver/Openshift-Apiserver/Management cluster Konnectivity agents: agent -> kube-service entry -> konnectivity server

The key thing to focus on is client network/external client network. Client’s have a variety of configurations they are normally interested in integrating their clusters with. Some tangible ones we have seen from customers (Satander, CitiBank, etc)
- Want all traffic from all Kubernetes nodes to flow **through a single dedicated proxy/egress box using HTTP_PROXY vars and also proxying tcp traffic**
- Want to IP whitelist set IP ranges on the load balancers fronting the API Server to control access.

The traffic controls on the management cluster side (load balancer/nodeports) are independent from the controls in the client network path but work together to form a full solution. Let’s start

with the typical controls on the management cluster side
1) IP whitelisting at any load balancer fronting API Server traffic:
- This can also be used for node port strategy if on backend a external load balancer just sends to nodeport ips
- Then fundamentally if using that: the kube cluster only allows ingress traffic from those nodeports from ips associated with the load balancer.

On the in cluster side: users control how they want kube-apiserver traffic to flow from the node by what they specify in the haproxy backend configuration. Some examples:

- Send direct to load management cluster ingress endpoints/load balancer (covered today in hypershift)
- Direct TCP traffic to a dedicated **proxy** box that all traffic in their network must flow through for security/analysis reasons. That proxy box then handles after analysis/approval forwarding the tcp traffic to the management cluster ingress endpoints. Typically one “proxy box” per network
- For Ignition/general HTTP traffic: teams have HTTP Proxies setup that must be utilized for all HTTP traffic (applications configured to use HTTP_PROXY variables).

IBM Cloud does this through custom ignition configs in hypershift today. The key input configuration parameters for kube-apiserver traffic are the following:
- HAProxy Listening IP (**must match advertised-address of API server)**
- HAProxy Listening Port (**must match secure-port of API server)**
- HAProxy Backend Address list
    - This is where the TCP streams are directed to. In the simplest case it is directly to the cluster ingress backends. In more complicated cases: this might point to the “proxy box” that analyzes traffic
    - In simple case this is the ingress endpoint DNS name + the node port the kube-apiserver is exposed on. Example: `c100-1.containers.test.cloud.ibm.com`:32167,`c100-2.containers.test.cloud.ibm.com`:32167,`c100-3.containers.test.cloud.ibm.com`:32167
- For HTTP Traffic on the node: typically the user configures “shell environment variables” at /etc/environment and default global systemd environment variables to get all systemd units to register proxy vars. For containerized traffic: IBM Cloud deploys a “bridge” called Satellite link where there’s a server deployment in IBMCloud and agents run in the containerized environment. The agent forms a connection to the server and using websockets (so it can integrate with HTTP proxies, etc) and then provides unique TCP endpoints in the SDN (Kubernetes services) that send to the local side of the “agent”, are tied to a unique websocket connection (which is ultimately tied to a unique DNS backend like IBM Cloud Object Storage). We provide APIs that allow users to specify what the endpoints are they want to proxy.

This key config variables are sent through a couple custom ignition config entries and then a script is utilized to take those variables and ultimately build a full haproxy conf shown here:

https://github.com/relyt0925/rhcos-vpcgen2-ignitiondata/blob/master/usr/local/bin/ibm-dynamic-master-haproxy-config.sh#L24-L35
https://github.com/relyt0925/rhcos-vpcgen2-ignitiondata/blob/master/usr/local/bin/ibm-dynamic-master-haproxy-config.sh#L8-L9
https://github.com/relyt0925/rhcos-vpcgen2-ignitiondata/blob/master/systemd-units/ibm-dynamic-master-haproxy-config.service#L8
https://github.com/relyt0925/rhcos-vpcgen2-ignitiondata/blob/master/etc/haproxytemplates/haproxy-ibm-master-endpoints.cfg#L29

Bring your own PKI Support

IBM Cloud has the following requirements that resulted in them needing to bring their own PKI generation logic and use the secrets hypershift define as the contract:

General Requriements:
1) Full CA Rotation Process: Critical for when an admin leaves an organization and was able to download admin certs at a period of time or if a successful attack was performed and someone was able to get the cluster CA. This process is defined here: https://cloud.ibm.com/docs/openshift?topic=openshift-security#cert-rotate

At a high level the process goes:
1) Generate new CA and add it to the CA bundle (in parallel with the old CA)
2) When new CA added: all certs issued going forward begin using the new CA
3) User guided through reloading all workers in their cluster (need new worker certs) and need to ensure they update all out of band automation (redownload any certs, update jenkins jobs, etc.)
4) When all those steps performed the user rotates the CA: which effectively revokes the old CA. At this point anything using the old CA would no longer to be authenticated

2) Additional domains necessary to be added to some certs
IBM Cloud has additional domains that need to be added to the exposed nodePort services (ignition, oauth, kube-apiserver, konnectivity) that are necessary based on the networking environments IBM Cloud supports. Some specific use cases:

For private + public endpoint clusters: Certs need to be signed with both endpoints. Currently: only one endpoint is signed (which is the one that is provided in the nodePort strategy)

For VPC Gen 2: There's VPE endpoints: https://cloud.ibm.com/docs/vpc?topic=vpc-about-vpe that provide domains that map to IPs within the users VPC that provide them access to their masters across the IBM Cloud VPC private network

For Satellite: There's satellite link domains that are additionally added to certs which are utilized to provide a domain that provides access from IBM Cloud into the Satellite location and ultimately to the exposed service

3) Auditable and controlled process for certificate refreshes
IBM Cloud is required to provide an auditable record for users when certificates are refreshed. Tangibly this means there need to be clear APIs and progress records that the users can refer back to to know when their certificates are refreshed. Note that certificates are only signed for a year so must be refreshed periodically in order to ensure the cluster is healthy.

Today we have clear API workflows that the user can prompt and then reference in audit log history (and provide to auditors) on when they have performed a specific action like a CA rotation. The timeline of certificates are refreshed whenever a user triggers an operation to the master (like a patch update or master refresh on the IBM Cloud side). These are also provided in audit logs that can be used for a user to reference back to when cert timelines in their cluster were refreshed last. It is hooked into alerting engines to know when a user is approaching expiration.

4) Need the ability to adopt clusters in with existing PKI (IBM ROKS Toolkit clusters)
We need the ability to adopt ROKS toolkit clusters that already have PKI generated and active in a given cluster. This PKI needs to be adopted and utilized in order to ensure production clusters are not broken and ultimately migrate entirely off the ROKS toolkit architecture.

CA Cert

Tugboat Architecture

Edge nodes

- Nodes run ingress components (specifically keepalived managing a floating IP in classic environments.
    - By ingress components I specifically mean any component necessary to route traffic from outside of cluster to the exposed multi tenant master node ports.
    - All traffic flows through the keepalived pods from the outside world
- 2 nodes per AZ (for failover using VRRP)
- All traffic from outside cluster flow through these nodes
- Since these process all ingress traffic, nothing else runs on them to ensure maximum bandwidth for network processing
- B3c.8x32.encrypted (8 core 32 gig RAM). Dedicated VSI (no noisy neighbors)
- 110 pods per node

Dedicated-Prometheus

- Per AZ node dedicated to running a Prometheus instance (only one zone active for prometheus at a time)
- Prometheus limits
    - 54 Gigs Memory
    - Has persistent storage that it uses to store prometheus data due to shear load and the fact that the retention policy is 30 days.
- B3c.16x64.encrypted Dedicated VSI (no noisy neighbors)
- 1 node per AZ
- 160 pods per node

Customer-workload

- Amount of nodes per zone relative to the number of customer masters being supported by the tugboat
    - Largest cluster: 116 per zone (us-south)
        - Scaled to support 440 openshift masters
    - Smallest cluster: 10 per zone (jp-osa)
        - Scale up monitored when memory/CPU in a zone get below threshold
    - Runs master components including etcd, kube-apiserver, openshift-apiserver, etc, etc. Only runs master components
- B3c.16x64.encrypted (16 core 64 gig RAM). Dedicated VSI (no noisy neighbors)
- 160 pods per node

Kube-Resources/General pool

- This is a general “fallback” pool where no taints are applied to the workers in the worker pool
- For general items that might not tolerate specific taints like kube-metrics-server, openvpn client (NOT customer openvpn client the one specifically for the tugboat allowing logs/exec, etc)
- 1 per zone
- B3c.8x32.encrypted Dedicated VSI (no noisy neighbors)
- 110 pods per node

18, November 2021

- Talk on Prem CoreOS strategy

11, November 2021

- Bringup security patch issue: https://github.com/openshift/hypershift/issues/414
- Question on machine-approver and need for bootstrapper cert?

    - Moved to func ReconcileKASMachineBootstrapClientCertSecret(secret, ca *corev1.Secret, ownerRef config.OwnerRef) error {
    - return reconcileSignedCert(secret, ca, ownerRef, "system:serviceaccount:openshift-machine-config-operator:node-bootstrapper", []string{"system:serviceaccounts:openshift-machine-config-operator", "system:serviceaccounts"}, X509DefaultUsage, X509UsageClientAuth)
- Customer questions on MCO in plans?

4, November 2021

- Talks on UPI bootstrap and upgrade architecture. Notes on UPI “cluster-api”

28, October 2021

- Backport 4.8
- In place bootstrap stuff

14, October 2021

- Should these CRD updates be being seamless? Example on recent
- If time: talk about way to disable operators/etc when cluster goes “unsupported” so unsupported cluster stays in steady state

07, October 2021

- Satellite RHCOS bootstrapping strategy

30, September 2021

- Satellite RHCOS bootstrapping strategy

23, September 2021

- Portieries PR followup

16, September 2021

- KMS pr follow up

9, September 2021

- [dan] Tyler proposed adding IBM cloud support to the e2e suite (https://coreos.slack.com/archives/C01C8502FMM/p1631111795320700), let's talk through what that would look like and how we can make it happen as we really need it
- Talk about some security ignition stuff like rotating token (just keep on radar)
- Talk about interesting satellite RHCOS “on prem” pooling plan

26, August 2021

19, August 2021

Figure out when to schedule KMS design review before implementation
Determine feasible strategy for end of September on if secure cert approver for

secure kubelet serving/client cert approving system:
https://github.com/openshift/hypershift/issues/279

Rotation of ignition token: https://github.com/openshift/hypershift/issues/364

Cleanup stale old nodePool ignition tokens/secrets:
https://github.com/openshift/hypershift/issues/411

RHCOS HA multi zone ignition server:
https://github.com/openshift/hypershift/issues/406

MVP: what are build processes for cluster api and cluster autoscaler images:
https://github.com/openshift/hypershift/issues/414

12, August 2021

No meeting
Figure out when to schedule KMS design review before implementation

4, August 2021

Quick runthrough of task assignment (look at some potential long running RHCOS ones too)

29, July 2021

Talk about removing logs issue:https://github.com/openshift/hypershift/issues/266
Could we look into caching clients/reducing load?

22, July 2021

Talk about removing logs issue:https://github.com/openshift/hypershift/issues/266
Could we look into caching clients/reducing load?
Ignition server strategy

15, July 2021

General talks about IBM's strategy

08, July 2021

General talks on remaining end of July items

01, July 2021

Are we deploying ROKS metrics: yes we are ultimately
Close secret leak from control plane to user clusters:
https://github.com/openshift/hypershift/issues/315

- Note shouldn't be passing IBM managed certs there: use per cluster certs on that one

24, June 2021

Discuss time to level set on design for other MVP features for IBM Cloud:
Specifically:

- Named Certs
- Secure Port override
-Network type specification
- Oauth Identity provider

Discuss timeline on when dnsmasq conf will be removed or if we need to build image locally for Staging Dev preview at end of July

17, June 2021

Discuss automation to rollout updates to ignition configuration for IBM Cloud.
Curious on why package server needs vpn and if there's plans to get that using proxy apis instead of vpn? (or a mode where we can run in cluster)
Also DNSMasq needs to be able to adjust to management control plane service ip range
Note (Friday meeting): Discuss Bring your own certs and etcd design

10, June 2021

TODO: discuss PRs for ibmcloud cluster CRD and apiserver webhook

Discuss "auto approver" strategy proposal

03, June 2021

TODO: discuss general code architecture for adding containers/config to kube apiserver
https://coreos.slack.com/archives/C01C8502FMM/p1622490832188000

Main takeaway: we can keep everything cloud specific in the platform spec for just ibm cloud. We should stay away from annotations unless it makes sense

27, May 2021

Went through grouped issues into general "architecture" categories and talked about how they would be solved
Discussed progress on centralized ignition server for scaling purposes. Have a plan for exposing one node port but currently still need a ignition server for every node pool. Will optimize that at a later time

20, May 2021

- [Tyler] 25 created issues: https://github.com/openshift/hypershift/pulls?q=is%3Aopen+is%3Apr+label%3Aarea%2FROKS
    -
- Changes that a user does during the life cycle of a cluster should not change the amount of ingress points for a given cluster
    - Ideal target for ingress points for the cluster <4 total per clusters
- IBM perspective: only one version being served by MCS at a given period of time and it corresponds to what the control plane is at version wise
    - Hopefully no need for multiple Ignition servers
- Next steps: team(s) get familiar with issues (read through them) and identify if there are any that are of interest to them and fit with their current schedule (taking into account other responsibilities/priorities)
    - Next meeting we will look to assign some of those and then identify what's remaining
- IBM performance team looking at scaling metrics associated with hypershift

13, May 2021

- Derek: Red Hat desires to ensure that IBM needs to build and release hypershift operator independent of OCP release builds. Ensure that RH team is not in the IBM operational loop/release for July-October timeframe.
    - Tyler: Agreed. Already planning to build and release hypershift-operator
- [Jake Kitchner]
    - Goal: We want to use **RHCOS the primary OS on IBM cloud**.
    - IBM Cloud is going FedRamp in 2022, goal is to enter audit period in Jan
        - Fully validated FIPS stack of stuff (we could figure out how to do that with existing RHEL 7 stack) but we want to make things consistent.
    - IBM Cloud satellite is also part of our goal (IBM's distributed cloud offering)
        - BYO infra and run IBM cloud services on that infra
        - Highlight RH OpenShift on IBM Cloud
            - We want to synchronize that with RHCOS in the future
- [Derek]
    - What are the basic infra goal for the July timeframe

    - We want to be able to enable you to get builds without waiting for RH to get you official image while still keeping the official image just accelerating the pace.
    - FIPS is also a bit tricky, needs a special compiler
    - Goal July-October.
- **Responsibility Matrix**
    - Konnectivity (IBM) - has stability issues but IBM has a networking tiger team to get down to issues and resolve them.
    - [Cesar] HyperShift tries to control the life-cycle of the CP NS vs ROKS (which does not)
        - CR to coordinate between different control (HyperShift and IBM's controllers).
        - [Tyler] system as is today works even if lay our namespace , Bring our Etcd,.. It works!
    - [Tyler] Calico (IBM)
        - We need to be able to send Calico instead of OpenShiftSDN.
    - [Derek] CNIs
        - NetworkType: None (RH)
        - CNIs - any third part CNI should be able to work.
    - [Tyler] OIDC IBM …
        - Expose a section for OIDC config
        - Can we specify more than one IdP?
            - [Cesar] Yes that should be no problem…
    - Cert generation
    - Multi-zone (running three replicas instead of having one replicas)

Audit webhook Writeup

Purpose
Audit webhooks allow the user to specify an endpoint to send API Server audit events to through a kubeconfig. This allows the endpoint to monitor and process these events to see if they detect any malicious/"outside the norm" activity and take the appropriate action on that.
Sorts of activity that can be detected/monitored:
Unexpected exec connections
Secret creations/retrievals
Node data deletions
etc

Very specifically in the case of IBM Cloud customers: there are two common use cases: One is

forwarding the audit logs to Log Analysis: https://cloud.ibm.com/docs/containers?topic=containers-health-audit. Teams can then view these logs overtime and generate queries/alerts based off specific activity. The more advanced use case used for compliance purposes is these logs are forwarded to QRADAR: https://www.ibm.com/security/security-intelligence/qradar . There is a dedicated team and policy in place that whenever certain actions are detected outside of approved IBM policy (execing into privileged containers, running nsenter, etc) that alerts are fired and teams through the notification process need to validate that the “Actions” are legitimate. There’s also the possibility for a customer to a customer managed endpoint that will then process the logs: https://cloud.ibm.com/docs/containers?topic=containers-health-audit#audit-api-server-external

Pros/Cons for generic across providers or specific to a provider
Exposed at a cloud provider agnostic level
Pros:

This is a base Kube/OCP feature at it’s foundation: the input to enablement is a kubeconfig that points to the desired endpoint. This is cloud agnostic and can be ran in any cloud. The only reason it wouldn’t be allowed is mainly from a “support” perspective and what the provider of the service wants to support in a given environment.

There is nothing fundamentally preventing other providers from adopting this. The ROKS team would also in the future like to utilize this in other platforms when we use hypershift in Satellite environments. Keeping it generic will allow seamless adoption without API changes.

Cons:
Openshift up to this point has specifically said it does not support customers configuring this. IBMCloud is the only provider that has allowed this to be exposed so far. If it’s platform agnostic: users might get the idea that they are allowed to use it even though the Openshift team hasn’t deemed they support this configuration in that customer’s environment like AWS.