Extension points:

- Policies
- Routes
- Services + Dao
- Credential types
- ~~ExpressJS custom middleware~~
- Models
- Admin
- HTTP and HTTPS servers custom events (like upgrade for websockets)
- CLI
- ~~Request\Response Transformation~~

Notes:

- plugins are mounted before processing gateway.config to have all policies available in config
- yaml config files can contain other sections, we are not restricting schema
- one plugin per package
- multiple policies, conditions etc. per plugin

Install

system.config.yml

```
plugins:
  'npm-plugin-name'
    prop1: test # settings for plugin
```

I would personally leave it the only way to install. Another variant is convention based in gateway.config.yml

```
policies:
  - test
```

will do `npm i express-gateway-plugin-test` I see the only problem is that we do not control these modules, they look like official supported, since anyone can publish with name express-gateway-plugin-bla

Uninstall

remove plugin definition, all app data will be preserved (do manual cleanup if needed)

Folder structure

I would not force any opinion on how people should structure code. developers can always do `require('./my-code')`

some plugins will be 10 lines and no need for folders at all, some will be with cross dependencies (let say policy to access service directly) so limited context in folder may not be enough. Documentation example can provide folder structure as reference

Policy

Interesting injection points in apigee, especially “post response” part where you execute something and not adding latency to client call. In our case logging could be much better in this way

http://docs.apigee.com/api-services/reference/javascript-policy

```
module.exports = function(manifest){
  // it can be explicit `manifest` or it can be just 'this'
  manifest.config // access to configuration {gateway, system, models}

  manifest.config.models.Roles = {
    properties: {
      name: {isRequired: true, isMutable: true},
      }
  } // some new model definition

  //New service declaration: oauth2 plugin will have code like:
  manifest.services.tokenService = {}

  // Existing Service: to avoid refactoring we can do decorators
  // Or Strategy if we have time to refactor existing services
  // (In this case main issue is to define common interface)
  manifest.credentialService = decorator(manifest.credentialService)
  decorator = function(credSrv){
    let prev = credSrv.getCredential
    credSrv.getCredential = function (id, type, options) {
      if(type === 'jwt'){ /* do jwt magic */ }
      else { return prev.apply(arguments) }
    }
  }

  //dao objects exposed as property of a service
  manifest.credentialService.dao.redis // this is lib/services/credentials/credential.dao.js
  // same trick with decorators to change behavior
  // the only problem with decorators vs strategy is that you can accidentally override other modules

  manifest.conditions.push(function(conditionManifest){})

  // policyManifest.params, policyManifest
  manifest.policies.push(function(policyManifest) { })
// define new gateway routes/middlewares
  manifest.gatewayExtensions.push(function(gatewayExpressInstance) {} )

// define new Admin routes/middlewares
  manifest.adminExtensions.push(function(adminExpressInstance) {})

  manifest.eventBus.on('serverReady', ({httpServer, httpsServer})=> {
    httpServer.on('upgrade', ()=>{}) // integrate websockets
    httpsServer.on('upgrade', ()=>{}) // integrate secure websockets
  })

// hotReloadManifest = {type: "gateway|system", config } // config is the new config
  manifest.eventBus.on('hot-reload', hotReloadManifest => {})
// command is {namespace, subcommands : [{filepath, namespace}] }
  manifest.cli.register(command) {}
}
```

Request\Response Transformation

```
 manifest.policies.push(function(policyManifest) {

       Return{
         policy: function (req, res, next) {
         },
         reqTransformations: [NodeTransformStream],
         resTransformations: [NodeTransformStream]
}
})
```

I suggest to apply reqTransformation right after pipeline was identified and resTransformations similar to:
https://github.com/koumoul-dev/node-http-proxy/commit/4db8736ed6d28018d5cb2f541370e6de0972d534