Pluggable indexes

Problem

Currently, index implementations are chosen in a statically defined way and it isn't possible for a Pinot distribution to override a choice of index made in the core open source codebase. Index implementations need to be both present and wired up in the Pinot codebase in order to be used.

This prevents several workflows:

1. **Feature incubation:** not all implemented ideas are good, and currently a user would need to contribute code to Pinot, or maintain a fork, to try an idea. Pluggable index implementations allows users to incubate ideas before contribution
2. **Highly specific features:** some features are good, but don't make sense for the OSS community and would create a maintenance burden. Highly specific use cases cannot be supported by a Pinot distribution without passing the maintenance burden onto the OSS community.
3. **Custom physical representation of data types:** being able to override an index implementation allows alternative physical representations of data types without the rest of Pinot needing to be aware.

Proposed Design

Allow users to register two factories via pinot-segment-spi:

- `IndexCreatorProvider`
- `IndexReaderProvider`

Default implementations of these interfaces will be provided within pinot-segment-local, via refactoring existing logic. The default `IndexCreatorProvider` will be taken from `SegmentColumnarIndexCreator` and the implementations of `IndexHandler`. The default implementation of `IndexReaderProvider` will be taken from `LoaderUtils` and `PhysicalColumnIndexContainer`.

Plugins can't define entirely new index interfaces because the rest of Pinot still needs to know how to construct and query the indexes, so the `Provider` interfaces will model existing Pinot index types. For example, methods will exist on each interface to create a new

`ForwardIndexCreator` (`newForwardIndexCreator`) and `ForwardIndexReader` (`newForwardIndexReader`) respectively, and separate methods (`new*IndexCreator`, `new*IndexReader`) to create other kinds of index creators and readers.

Users will generally not wish to override construction or reader creation of every index type and should not be required to duplicate the default logic. Moreover, when new kinds of indexes are added to Pinot in the future, meaning new methods are added to `IndexCreatorProvider` and `IndexReaderProvider`, source compatibility of plugins should be maintained. Therefore, a class inheritance based API is proposed.

It is not anticipated that there will be more than one agent wishing to override index implementations within a single process, so only one registration of an override is permitted. Override registration must take place before first use.