BUILDING A CLOUD PLATFORM USING NETFLIX OSS

CARL QUINN

About Me

- Managed Netflix Cloud Tools 4 years
- Software Architect at Riot Games
- Helping “Cloudify things”
- Sharing my experience to help others

About Riot Games

- Founded 2006
- League of Legends launched 2009
- About 1600 employees worldwide
- Dozens of datacenters worldwide

Multiplayer Online Battle Arena

League of Legends

eSports

League of Legends

LEAGUE OF LEGENDS STATS

67MILLION

MONTHLY ACTIVE PLAYERS

27MILLION

DAILY ACTIVE PLAYERS

7.5MILLION

PEAK CONCURRENT PLAYERS

STATS RELEASED JANUARY 2014

Phase 0: Context

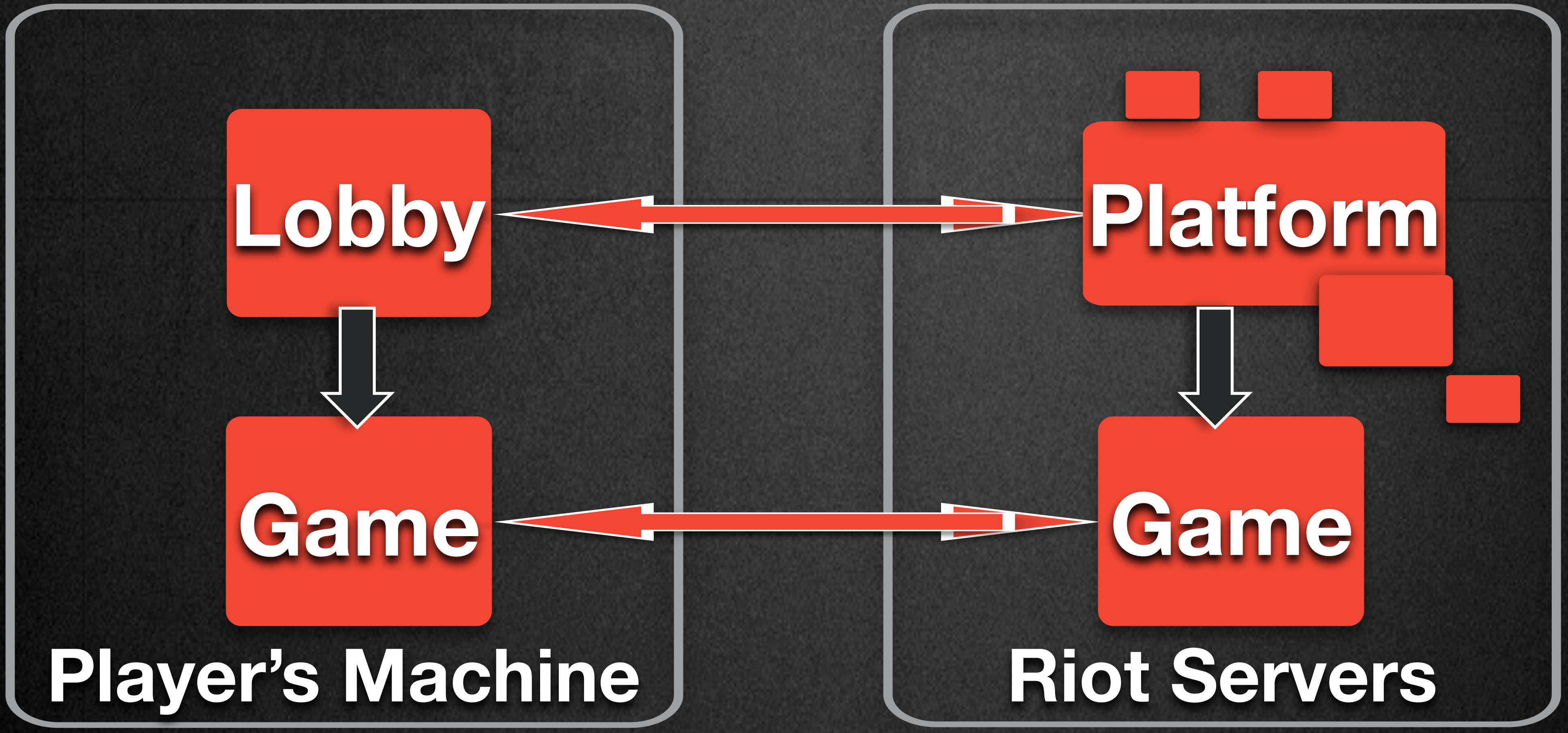
Engineering and Systems

Riot Engineering

- Independent product teams
- Favors independence over centralization
- AWS accounts not shared as much as at Netflix
- Different AWS uses and models

Riot Software

- Website: PHP
- Game lobby client: Adobe Air
- Game Platform backend: big Java enterprise app
- Other game services: Java
- Game client and server: C++



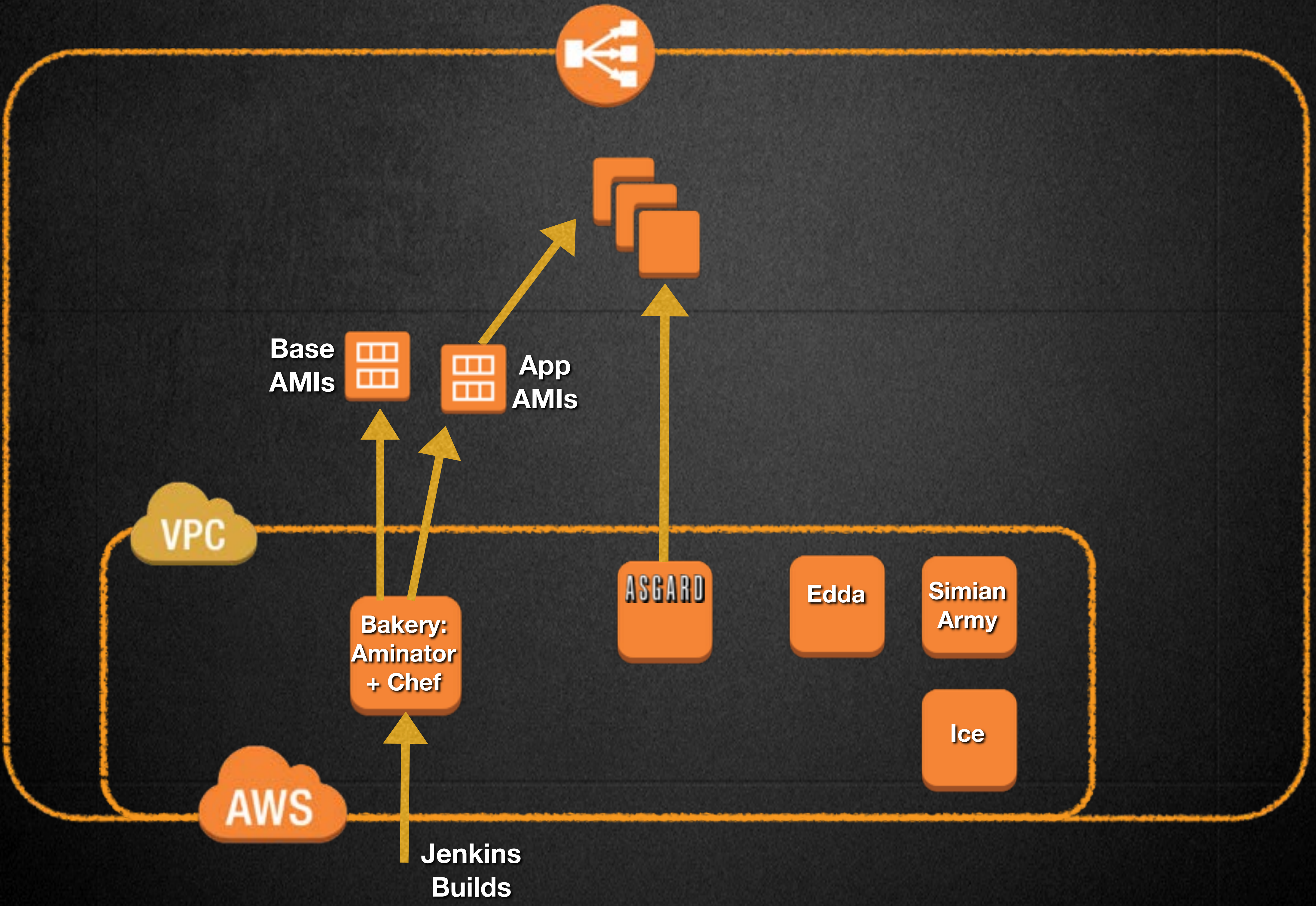
LoL Architecture

First Cloud Approach

- Quite a few teams already using AWS
  - Web site
  - API
  - Big Data
- First tasks
  - Standardize AWS build, deployment and management

Phase 1: Infrastructure

Netflix OSS Tools



Netflix OSS Cloud Tools

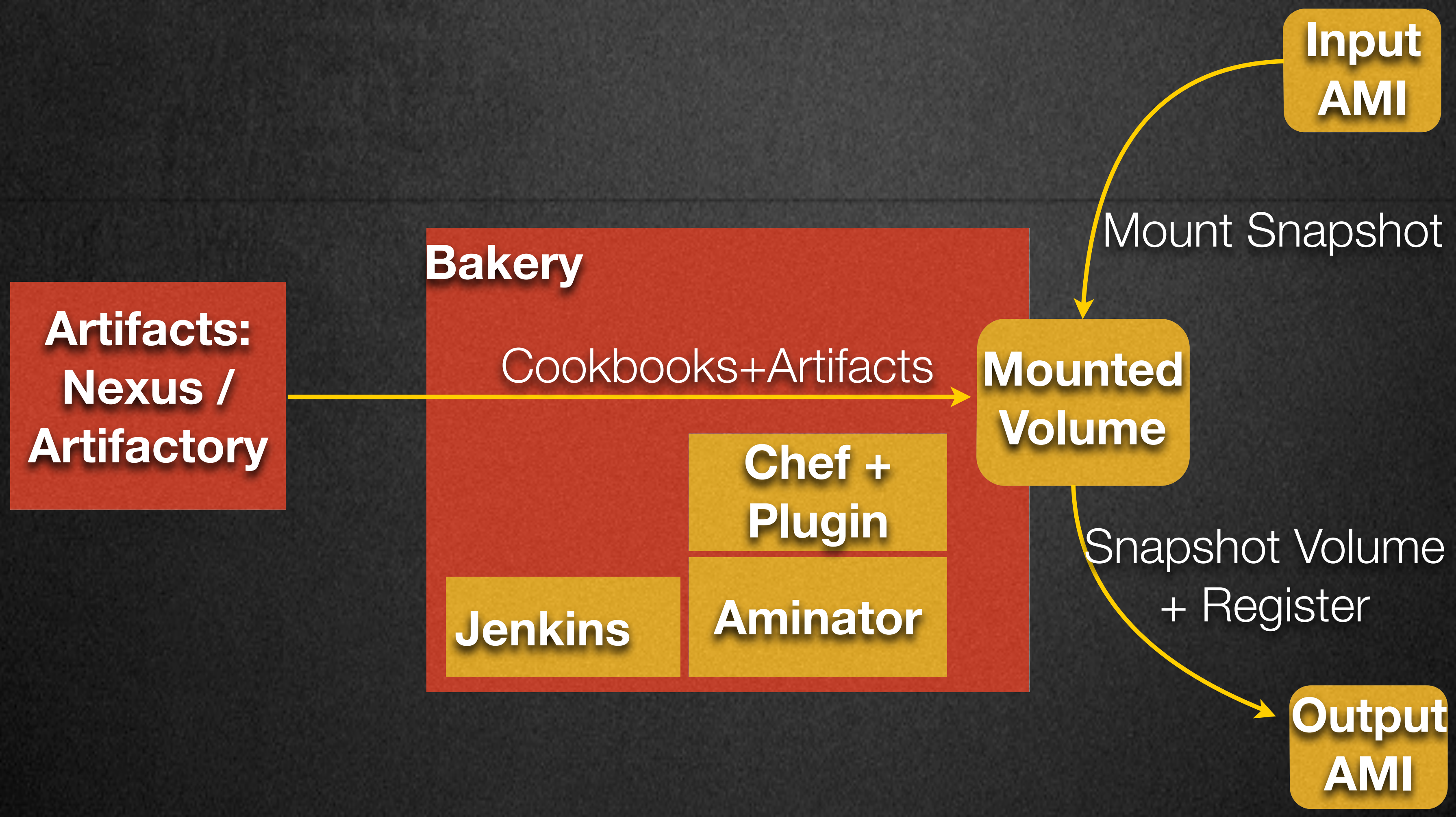
Build Pipeline: Bakery

- Added a chef-solo plugin to Aminator
  - Plugins are in: https://github.com/aminator-plugins

- Added Nexus auth

- Want to bake
  - Base and App AMIs
  - Ubuntu, CentOS & Amazon Linux

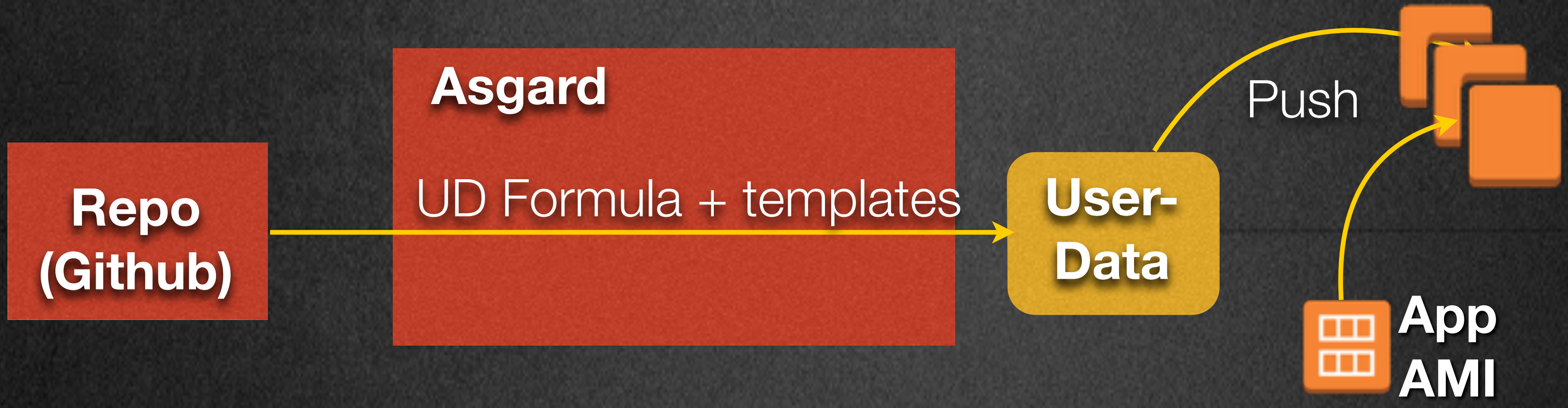
- Other options now include Packer



Baking with Chef

Deployment: Asgard

- User-Data: new plugin to use templates from Github
- Authentication: using Asgard built-in OneLogin support
- Authorization: TBD, less urgent w/ per-team accounts



```
plugin {
   advancedUserDataProvider = 'repoSourcedUserDataProvider'
cloud {
    repoSourcedUserData {
        base = 'https://gh.riotgames.com/api/v3/repos/rcloud/configs/contents/'
        suffix = '?access_token=0572f28db13e93433f68394faXXXXXXXXXXX'
        accept = 'application/vnd.github.VERSION.raw'
        formulaPaths = ['${app}/userdata.formula', 'base/userdata.formula']
```

```
format: shell
parts:
  - source: ‘base/global.properties'
    subst: true
    kind: copyTo
    target: /apps/${app}/conf/${app}-${env}.properties
```

Deploying with Asgard

Management

- Simian Army
  - Janitor and Conformity monkeys keep things tidy
- Edda
  - Client side scripting scans for security risk errors
- Ice
  - Provides visibility into usage by account & resource
  - Latest version has support for tag-based rollups

Great Success!

- Worked for smaller self-contained apps / services
    - Honu, API, etc.

- Projects already cloud-ready just get easier

But...

- Public Clouds don’t yet meet the needs of all of our apps
  - Regional Locality
  - Latency to Players
  - Partner Operations

- We still need (and like) datacenters

And...

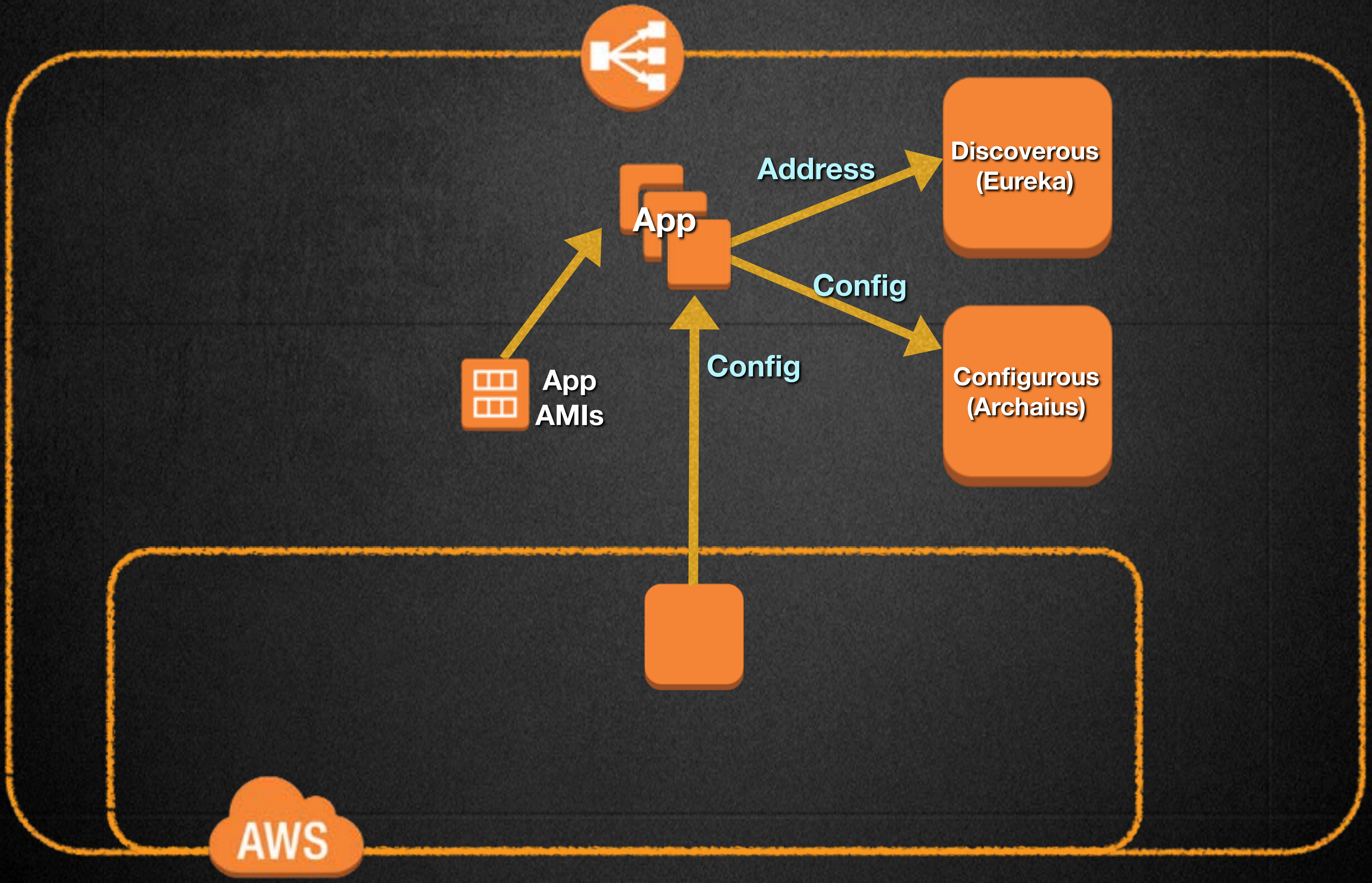
- Some projects were not ready for the cloud
- Need to break big apps into smaller REST services: SOA
- I.e. cloud-ify apps, making them cloud-ready
- Maybe going directly to AWS is not the best first step
- This is likely the case for many projects at many companies

Phase 2: The Stack

Netflix OSS Platform Libraries and Services

Join forces

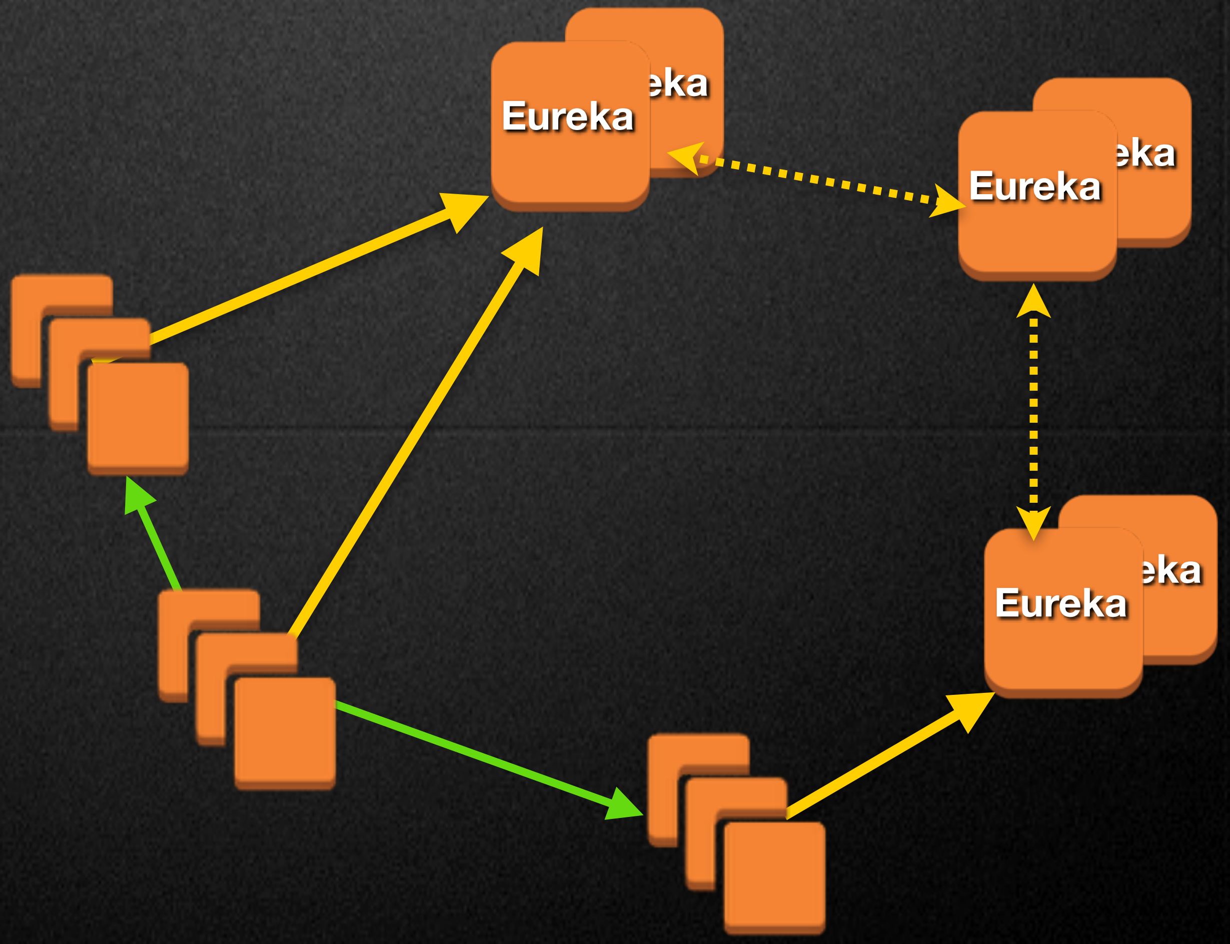
- Ongoing improvement initiative refactoring the Platform
- Our Ambassador spec:
  - Just REST, JSON, Swagger
- Our Hermes library
  - Jersey, Jackson
  - Dynamic Swagger generation
- Build on this to make existing services cloud-ready



Netflix OSS Cloud Platform

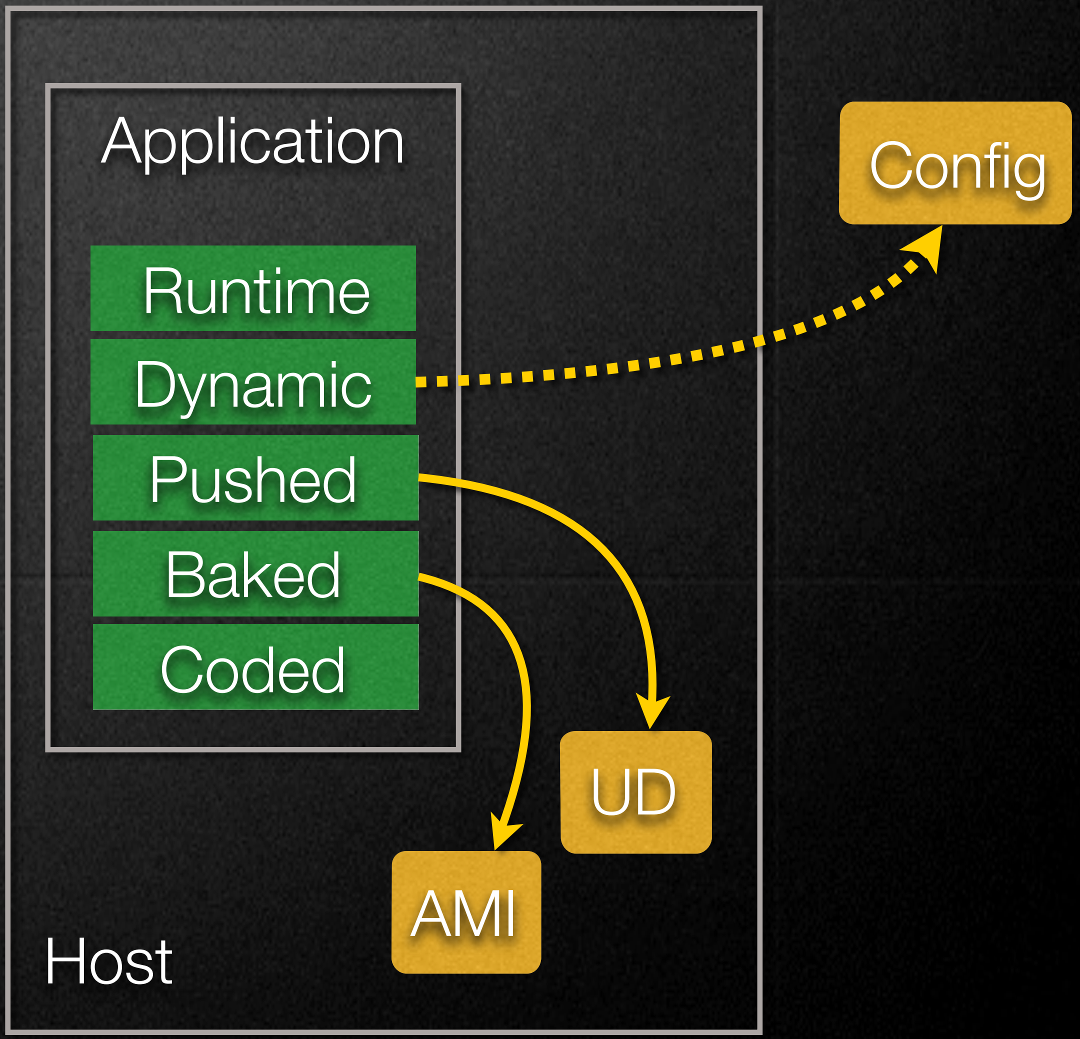
Eureka: a Discovery Service

- Server: Eureka-server can be deployed & clustered right out of the box
- Eureka-client is the client-side service address resolver.
- Riot variation on Eureka: Discoverous



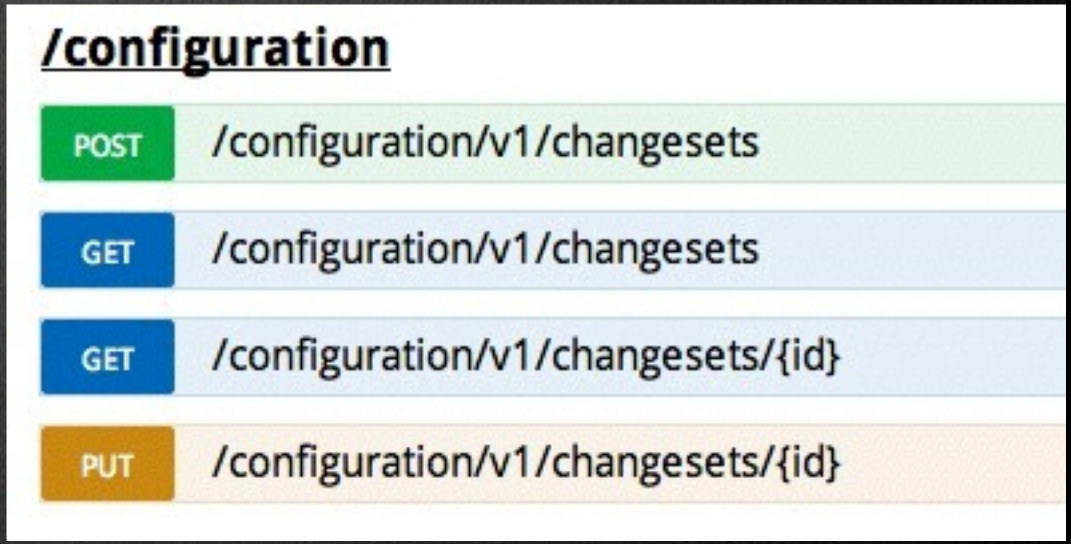
Archaius: Dynamic Config Library

- Aggregates multiple property sources
- Composites them
- Responds to updates
- Sources can be remote



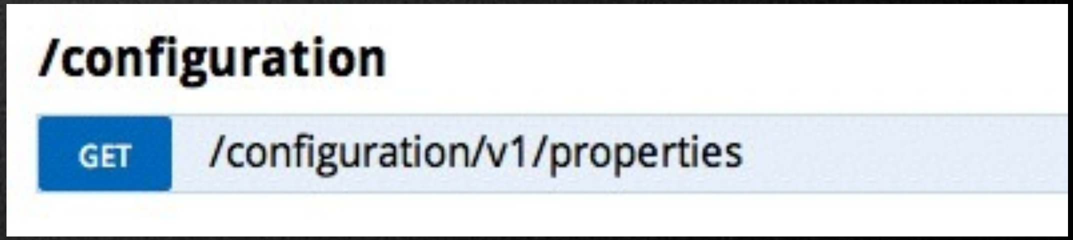
Configurous: our Archaius Service

- Java REST service with backing store (MySQL, RDS)
- Requests served from memory
- Provides property maps to app clients
- Provides transactional configuration sets to management front-end
- Planned to be open sourced



/configuration

POST	/configuration/v1/changesets
GET	/configuration/v1/changesets
GET	/configuration/v1/changesets/{id}
PUT	/configuration/v1/changesets/{id}

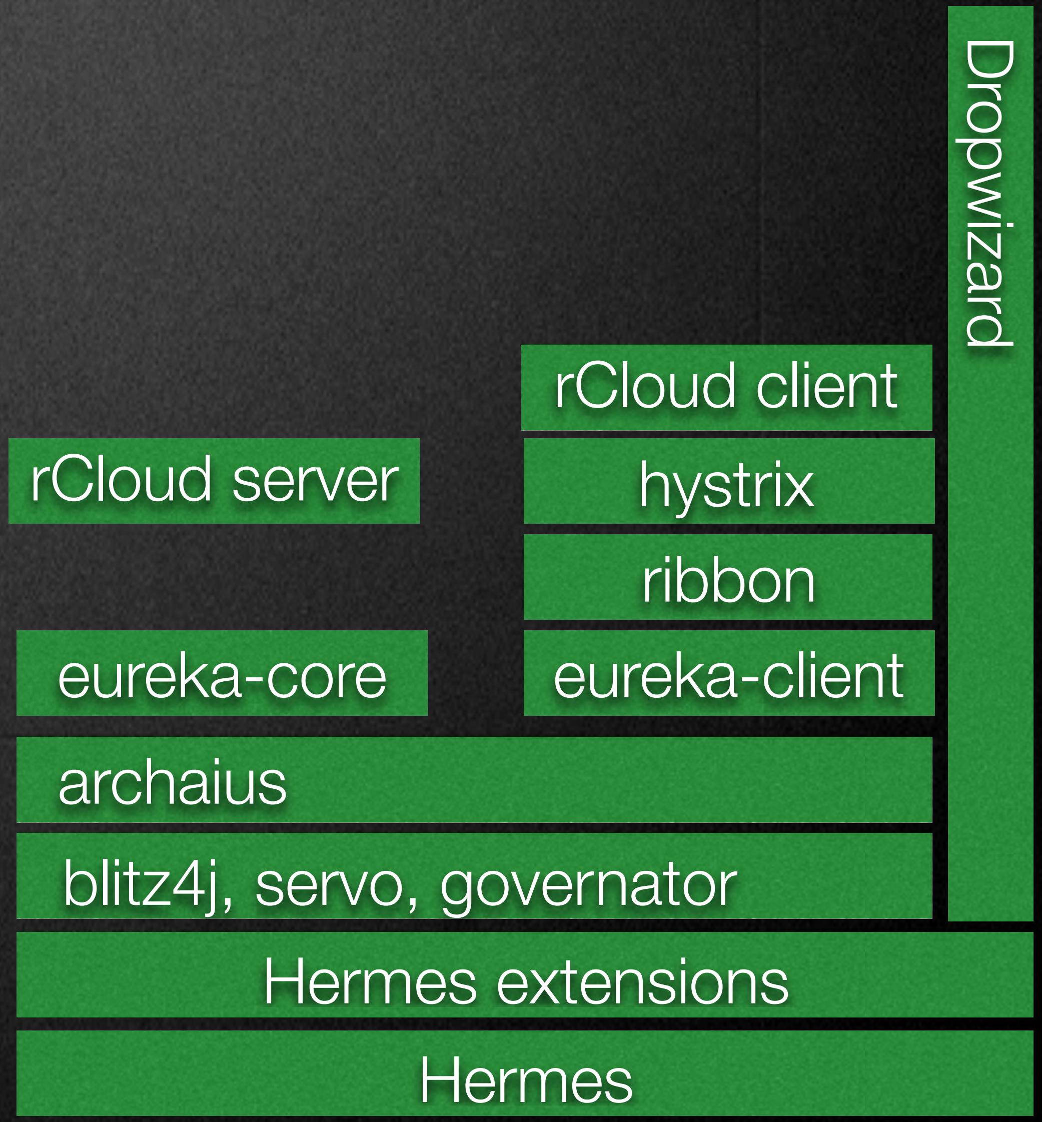


/configuration

GET	/configuration/v1/properties

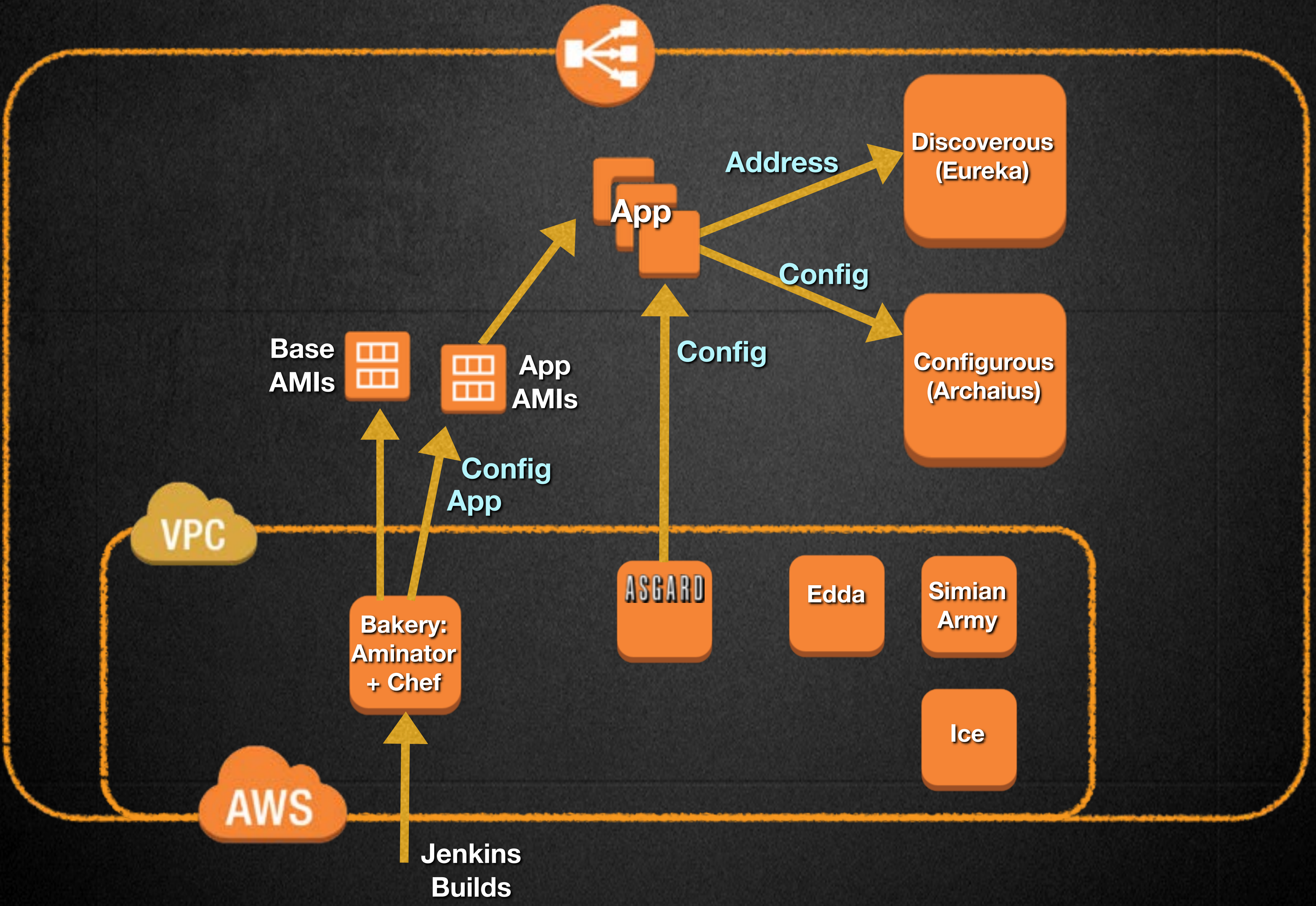
Hermes and rCloud App Kit

- Dropwizard
- Eureka-client
- Hystrix
- Ribbon
- Archaius
- Blitz4j, Servo, Governator
- Riot Hermes libraries



Phase 3: All Together Now

Deployment + Stack



Bake, Deploy and Run

Achievements

- Deployable Artifact for AWS
- Deployment Tool for AWS
- SOA Platform Infrastructure for apps
- SOA Platform Libraries for Java

Challenges

- Deployable Artifact is ***only*** for AWS, not Universal
- Deployable Artifact is kinda huge (10GB)
- Deployment Tool is only for AWS, not Universal

Phase 3: And Beyond

Universal System

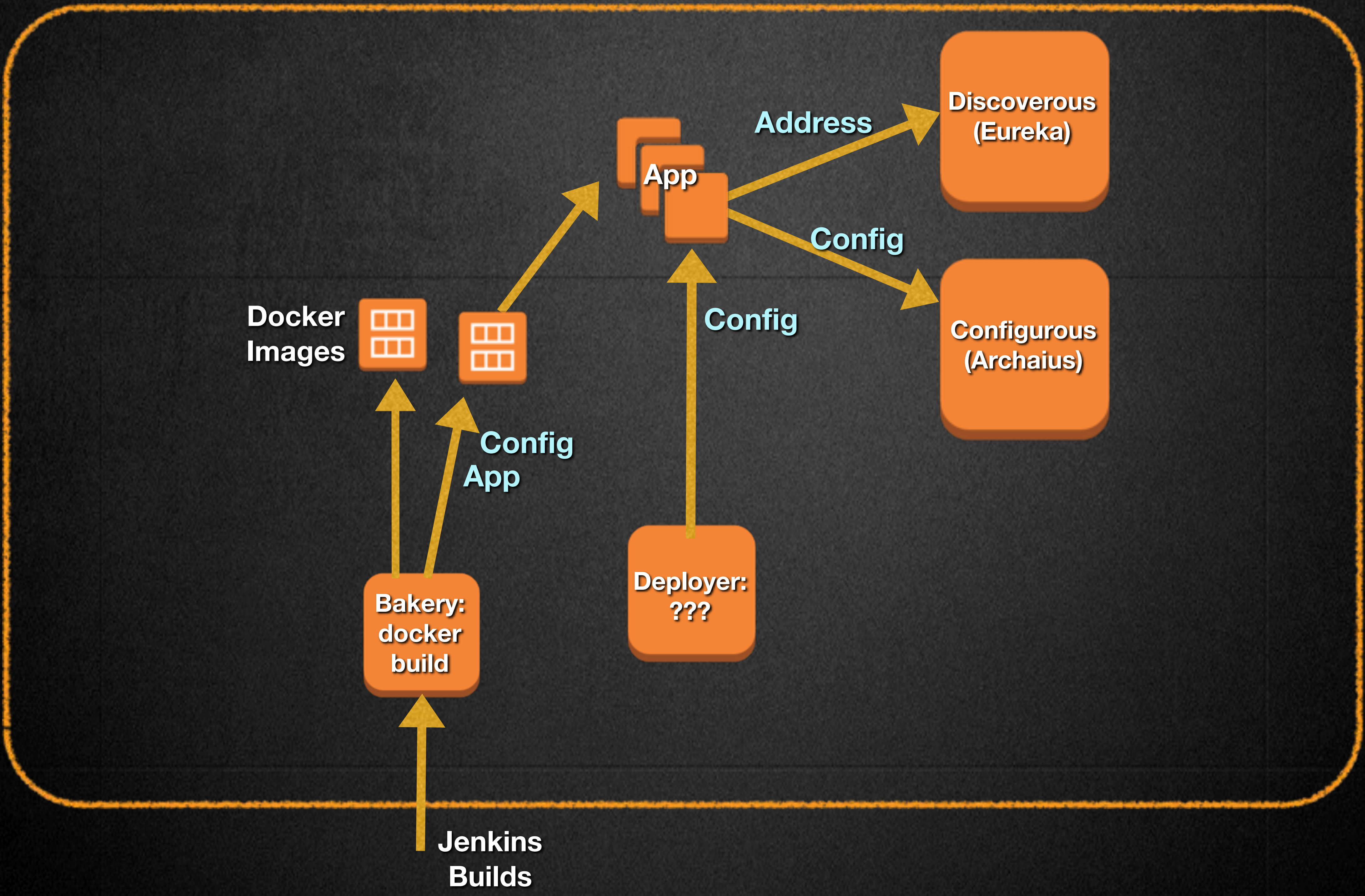
Grand Unified Future?

- Build application containers instead of whole machines
  - Dockerfiles and Docker images!
  - Universally portable
  - Small

- Deployer injects deployment config, orchestrates deployments
  - Fleet? Flynn? Mesos?

- Eureka (Discoverous) provides runtime service discovery

- Archaius (+ Configurous) provides dynamic configuration override



Bake, Deploy and Run, v2

Slides and Code

- http://parleys.com/play/516d6c9ee4b0a97ba5d91c71/chapter1/about
- http://www.slideshare.net/carleq/building-cloudtoolsfornetflixcode-mash2012
- https://github.com/aminator-plugins
- https://github.com/cyan-phi/chef-solo-provisioner/tree/add-secrets-with-local
- https://github.com/cquinn/asgard