Webinar

Continuous Delivery for Kubernetes Apps with Helm and ChartMuseum

Josh Dolitsky
Software Engineer
Codefresh

Stef Arnold
Sr. Software Engineer
SUSE

CHARTMUSEUM

Webinar

Continuous Delivery for Kubernetes Apps with Helm and ChartMuseum

Josh Dolitsky
Software Engineer
Codefresh

Stef Arnold
Sr. Software Engineer
SUSE

Outline

1. Intro to Helm
2. Helm Commands
3. Intro to ChartMuseum
4. ChartMuseum functions
5. CI/CD Pipeline
6. SUSE + Codefresh = <3
7. Demo

What is Helm?

- Helm is the package manager for Kubernetes
- Equivalent to “yum install <package>”
- Kubernetes manifest templates, packaged and versioned, referred to as **charts**

```
sh-3.2$ helm search stable/
NAME                                VERSION DESCRIPTION
stable/acs-engine-autoscaler        2.1.1   Scales worker nodes within agent pools
stable/aerospike                    0.1.5   A Helm chart for Aerospike in Kubernetes
stable/artifactory                  6.2.5   Universal Repository Manager supporting all maj...
stable/aws-cluster-autoscaler       0.3.2   Scales worker nodes within autoscaling groups.
stable/buildkite                    0.2.0   Agent for Buildkite
stable/centrifugo                   2.0.0   Centrifugo is a real-time messaging server.
stable/chaoskube                    0.6.2   Chaoskube periodically kills random pods in you...
stable/chronograf                   0.4.0   Open-source web application written in Go and R...
stable/cluster-autoscaler           0.4.0   Scales worker nodes within autoscaling groups.
stable/cockroachdb                  0.5.4   CockroachDB is a scalable, survivable, strongly...
```

Helm Use Cases

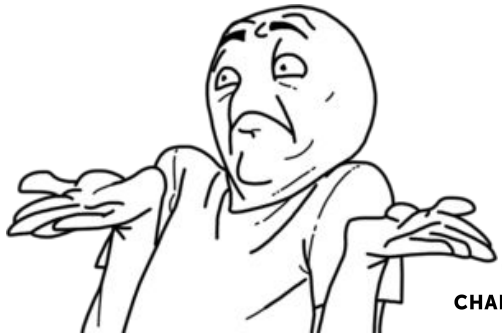
- Like other package managers Helm manages packages and their dependencies, and their installation.
- fetch, search, lint, and package are available client-side for authoring charts
- List, install, upgrade, delete, rollback for operations (makes use of server component Tiller)

Helm Use Cases

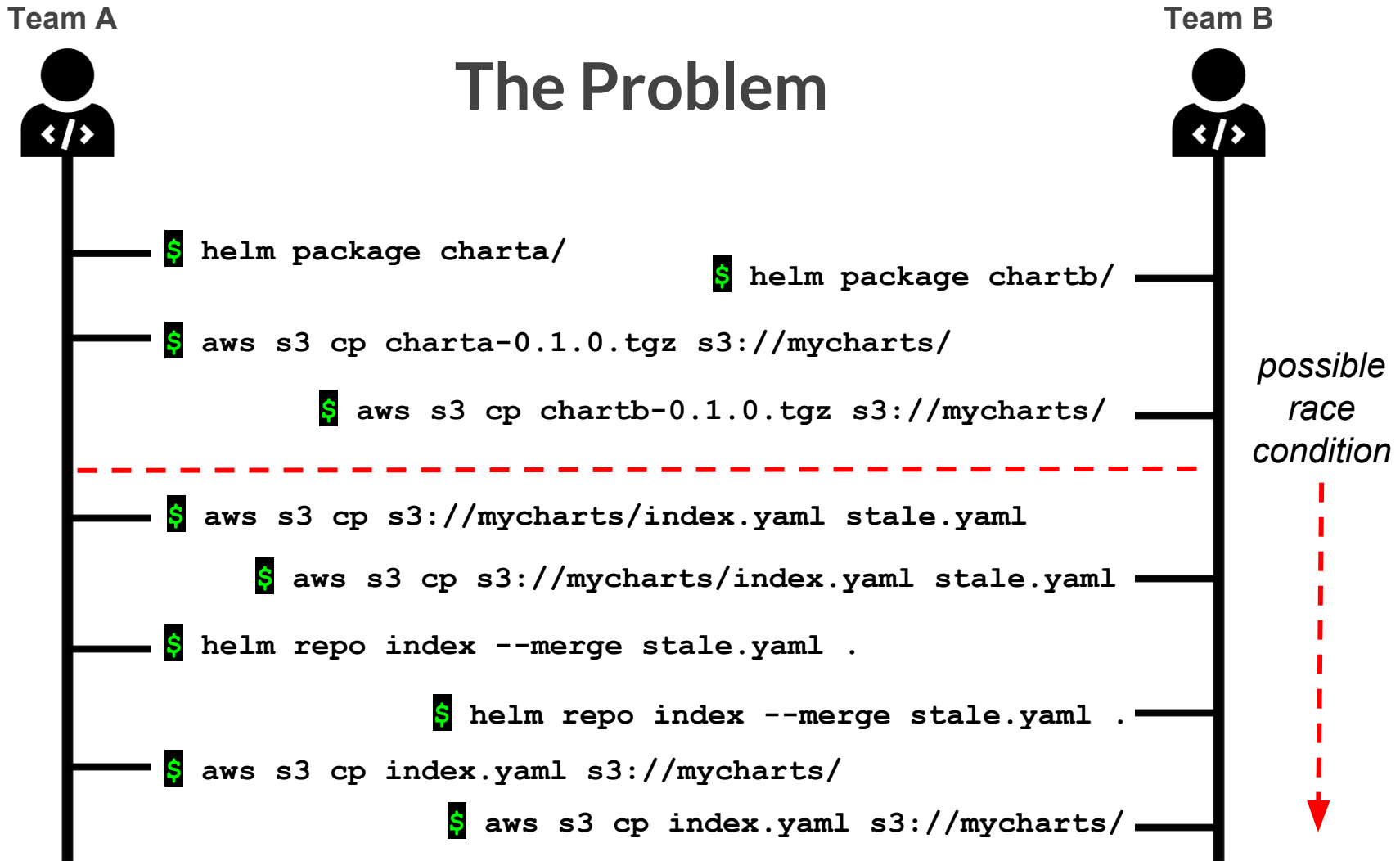
- Where do the packages live?
- What is a Helm repository anyway? index.yaml!

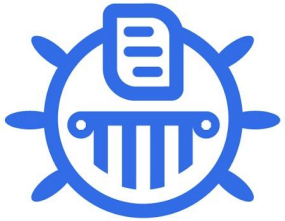
What's the problem?

How do multiple teams/devs publish their charts to a single repository *at the same time*?



CHARTMUSEUM


Team A
Team B
The Problem
$ helm package charta/
$ helm package chartb/
$ aws s3 cp charta-0.1.0.tgz s3://mycharts/
$ aws s3 cp chartb-0.1.0.tgz s3://mycharts/
possible race condition
$ aws s3 cp s3://mycharts/index.yaml stale.yaml
$ aws s3 cp s3://mycharts/index.yaml stale.yaml
$ helm repo index --merge stale.yaml .
$ helm repo index --merge stale.yaml .
$ aws s3 cp index.yaml s3://mycharts/
$ aws s3 cp index.yaml s3://mycharts/



CHARTMUSEUM

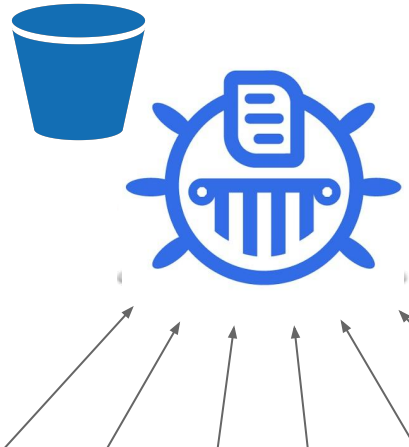
The Solution

Team A

```
$ helm package charta/
$ aws s3 cp charta-0.1.0.tgz s3://mycharts/
```

Team B

```
$ helm package chartb/
$ aws s3 cp chartb-0.1.0.tgz s3://mycharts/
```



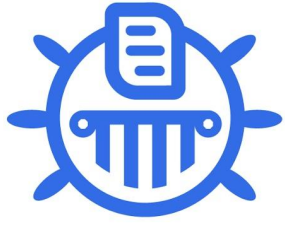
CHARTMUSEUM

https://github.com/kubernetes-helm/chartmuseum

README.md

ChartMuseum

PASSED go report A godoc reference *"Preserve your precious artifacts... in the cloud!"*



ChartMuseum is an open-source **Helm Chart Repository** written in Go (Golang), with support for cloud storage backends, including Google Cloud Storage, Amazon S3, Microsoft Azure Blob Storage, Alibaba Cloud OSS Storage and Openstack Object Storage.

Works as a valid Helm Chart Repository, and also provides an API for uploading new chart packages to storage etc.

Powered by some great Go technology:

- Kubernetes Helm - for working with charts, generating repository index
- Gin Web Framework - for HTTP routing
- cli - for command line option parsing
- zap - for logging

ChartMuseum: A chart repository server that adheres to “The Chart Repository Guide”

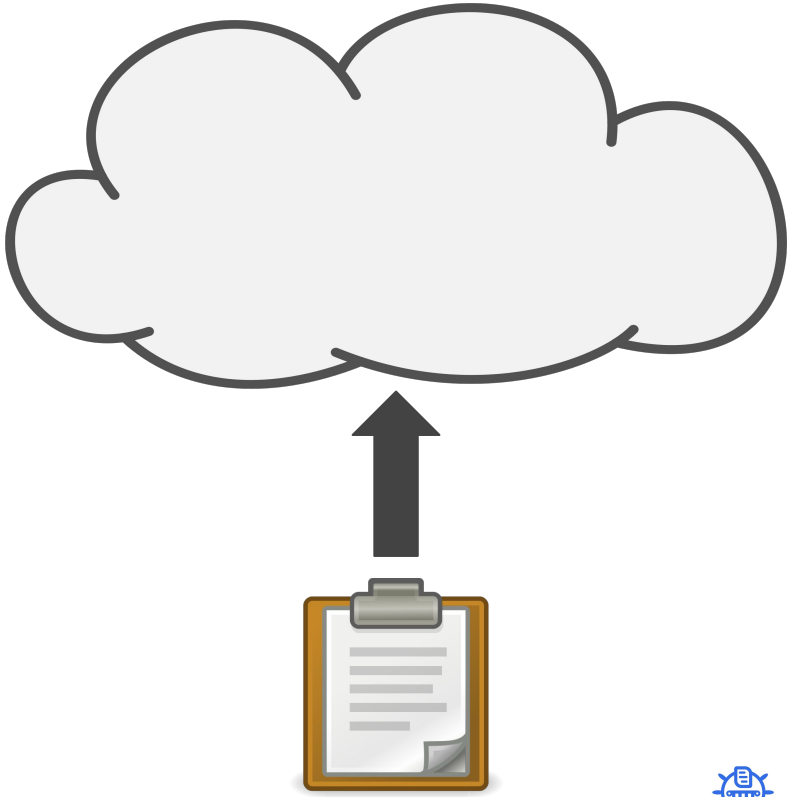
- HTTP server that houses an **index.yaml** file
- Serves packaged charts (.tgz)
- Provenance files (.prov) stored alongside chart packages

https://docs.helm.sh/developing_charts/#the-chart-repository-guide

CHARTMUSEUM

Features - Multiple storage options

- Local filesystem
- Amazon S3 (and Minio)
- Google Cloud Storage
- Microsoft Azure Blob Storage
- Alibaba Cloud OSS Storage
- Openstack Object Storage



CHARTMUSEUM

Features - API for uploading charts etc.

- `POST /api/charts` - upload a new chart version
- `DELETE /api/charts/<name>/<version>`
- `GET /api/charts`
- `GET /api/charts/<name>`
- `GET /api/charts/<name>/<version>`

CHARTMUSEUM

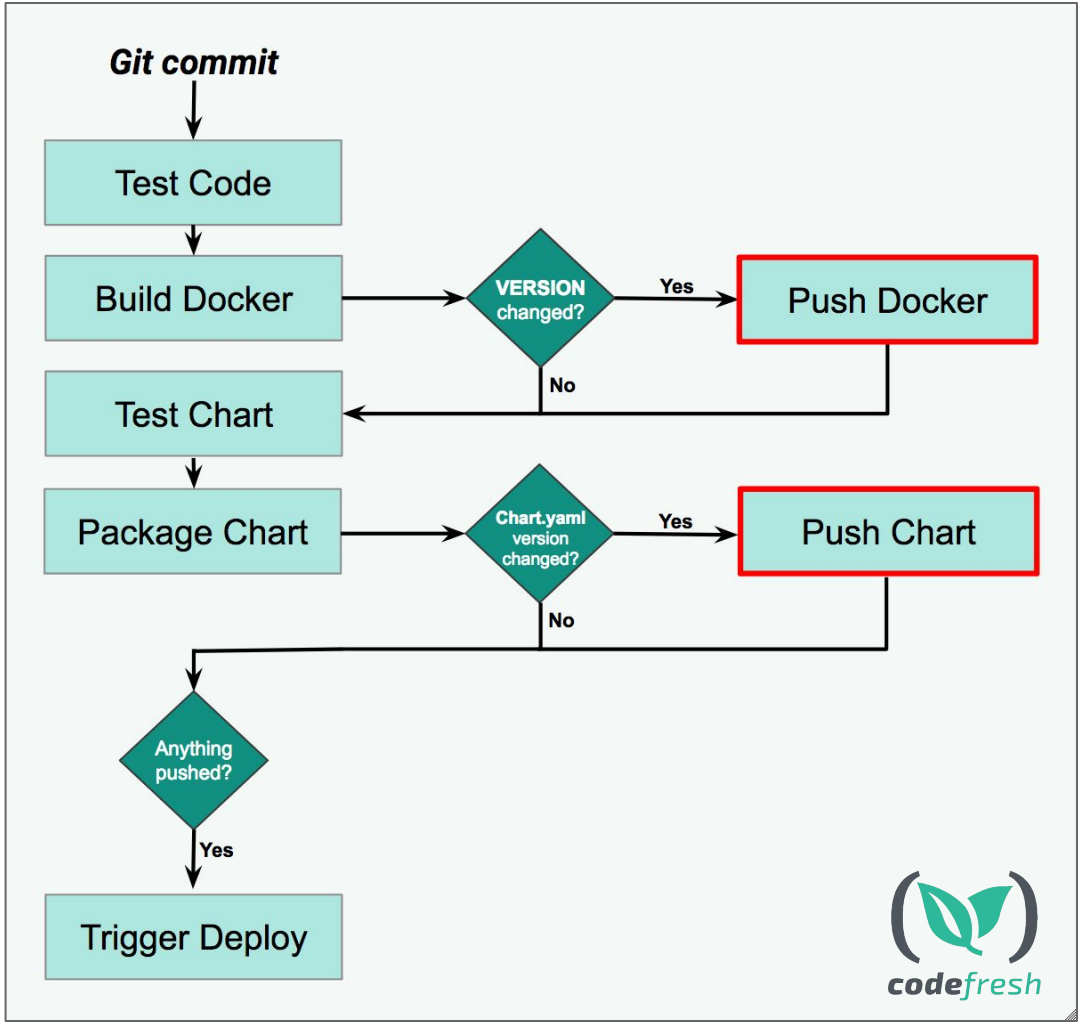
To the command line...

Deployments

We have seen how to:

1. Run a chart server
2. Author and package a chart
3. Upload a chart

Let’s look at a diagram...



Plug it into Codefresh!

Express our workflow as CF pipeline

First, we need a Kubernetes cluster (setup step walk through)

After, free form demo within CF UI

CHARTMUSEUM

SUSE

CaaS Platform

SUSE CaaS Platform allows you to provision, manage, and scale container-based applications.

It automates your tedious management tasks allowing you to focus on development and writing apps to meet business goals.

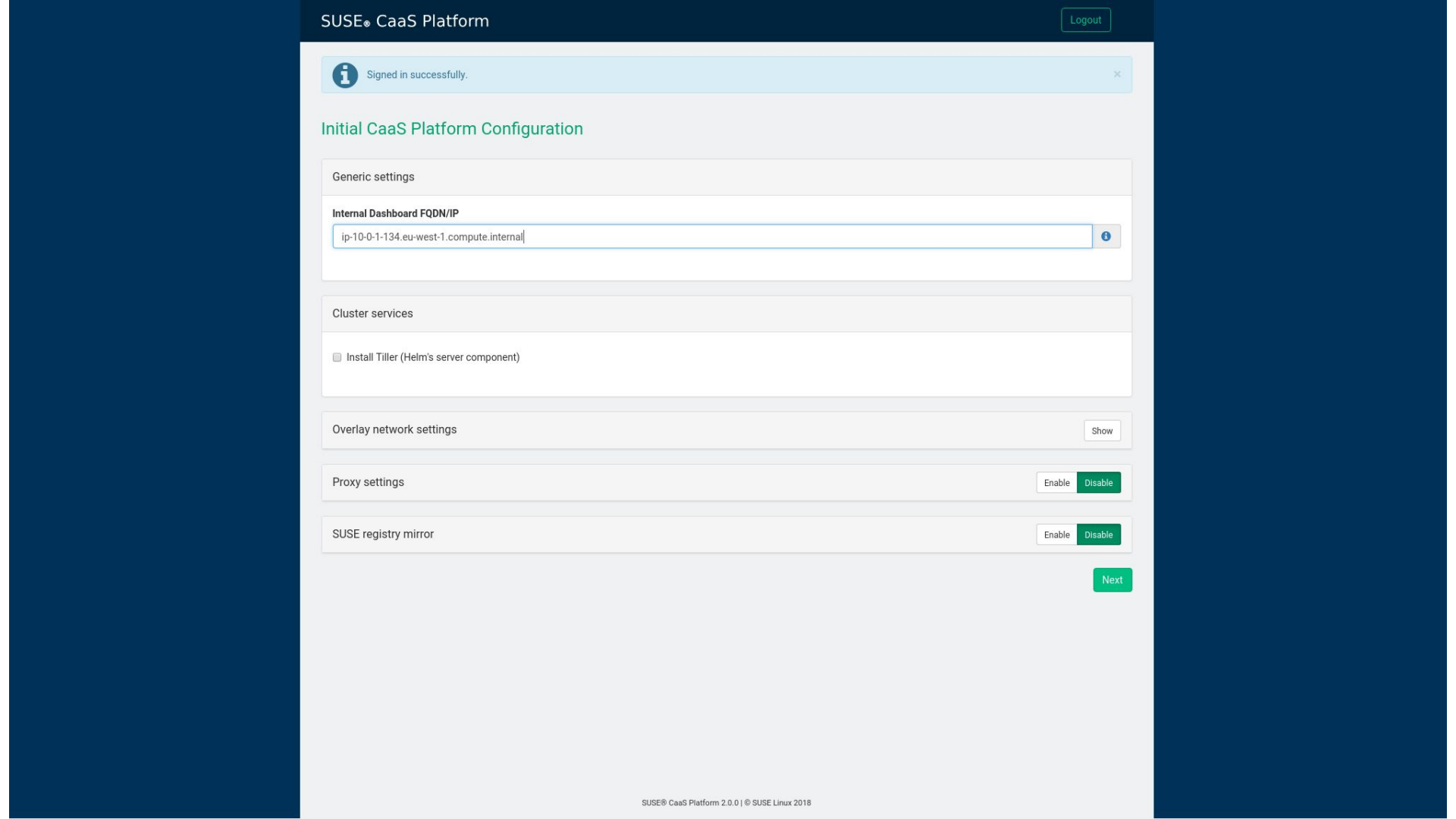
Log In

Enter your email address

Enter your password

Log in

Remember me


SUSE® CaaS Platform
Logout
Signed in successfully.
Initial CaaS Platform Configuration
Generic settings
Internal Dashboard FQDN/IP
ip-10-0-1-134.eu-west-1.compute.internal
Cluster services
Install Tiller (Helm's server component)
Overlay network settings
Show
Proxy settings
Enable
Disable
SUSE registry mirror
Enable
Disable
Next
SUSE® CaaS Platform 2.0.0 | © SUSE Linux 2018

Bootstrap your CaaS Platform in Amazon Web Services' Elastic Compute Cloud

In order to complete the installation, it is necessary to bootstrap a few additional nodes, those will be the Kubernetes Master and Workers.

Instance Type

General Purpose: T2
t2.xlarge

General Purpose: M4
m4.xlarge

Compute Optimized: C4
c4.2xlarge

Memory Optimized: R3
r3.xlarge

Storage Optimized: I3
i3.xlarge

Storage Optimized: D2
d2.xlarge

Other types...

General Purpose: T2

t2.xlarge

T2 instances are Burstable Performance Instances that provide a baseline level of CPU performance with the ability to burst above the baseline. The baseline performance and ability to burst are governed by CPU Credits.

Burstable CPU | Lowest cost

vCPUs
4 cores
RAM
16.0 GiB
Storage
EBS-only
CPU Credits / hour
54

Tip

Not sure which type of instance to use? Check the Instance Types list.

Cluster size

Number of instances

3

Total vCPUs
12

Total RAM
48.0 GiB

Tip

At least three nodes are required for a reliable cluster.

Networking

Subnet ID
subnet-e4fe97ad

Security Group ID
sg-9deb31e0

Questions

Josh Dolitsky
Software Engineer
Codefresh

Stef Arnold
Sr. Software Engineer
SUSE

CHARTMUSEUM

Relevant Links

- https://github.com/stefarnold/hello-cf
- chartmuseum.com
- codefresh.io
- suse.com/solutions/kubernetes/