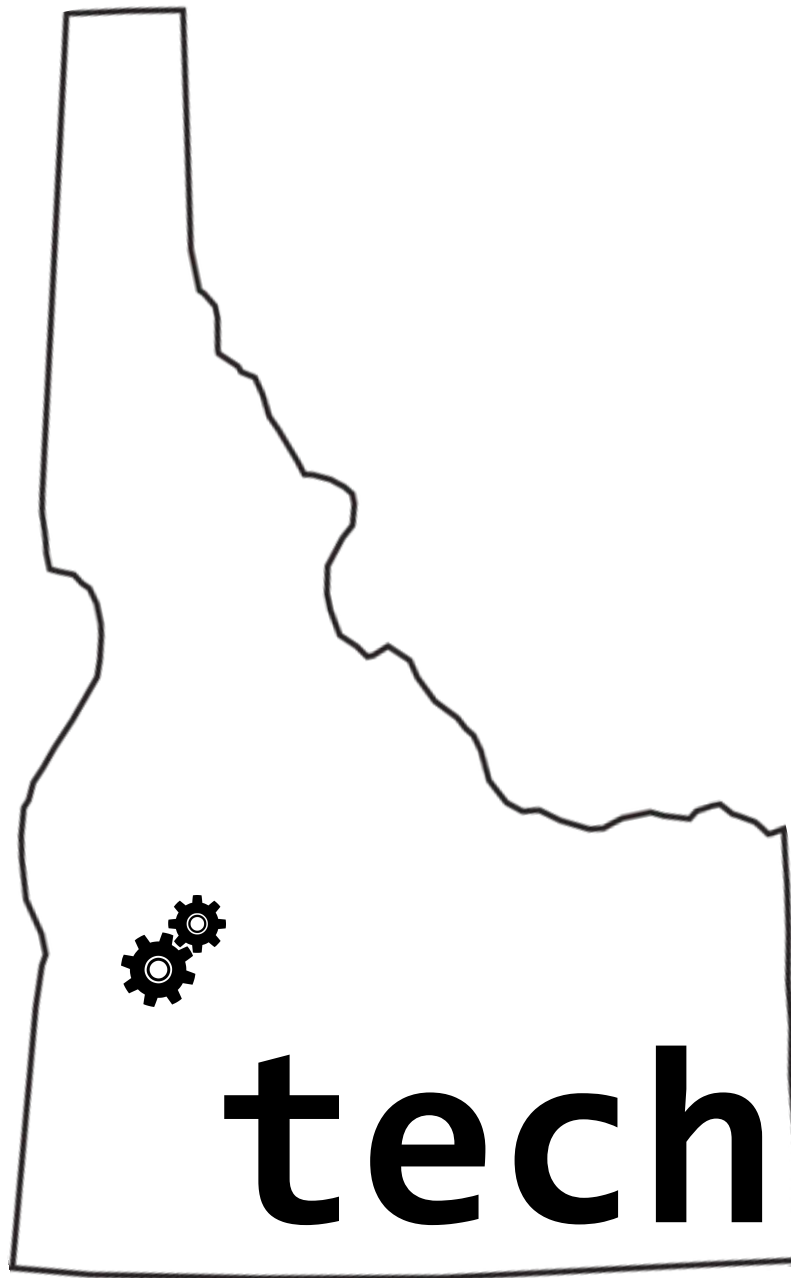


Cradlepoint's Journey to Kubernetes

Matt Messinger
Distinguished Engineer

@BoiseMatt





Kubernetes!





NetCloud Manager

Network Management | Analytics | Security



Branch Networks

SD-WAN | Edge Security | WLAN



IoT Networks

Internet of Things | Edge



Mobile Networks

SD-WAN | WiFi | Telematics

Industry leader in 4G/LTE network solutions and moving to 5G

NetCloud Manager - the early days.

2013 - 2015

The problem with success.

2016 - 2017

The problem of flexibility.

Too many ways to deploy code

Service(s)	CI	Artifact	Deployment Scripts (master → AWS)	OS Config on EC2 (app depend.)	Service Install on EC2	App Config on EC2 (environment & secrets)
A1, A2	Docker	debian	"stack-builder" tool	Chef	Native	Chef
B1	Docker	Docker & jar	custom bash + CloudFormation	Chef	Native	Chef
C1, C2, C3, C4, C5, C6	Docker	Docker	custom bash + CloudFormation	Salt	Docker	Consul/Vault
D1, D2	Docker	Docker & jar	custom bash + CloudFormation	Chef	Native	Chef
E1, E2, E3, E4, E5	Docker	Docker & deb	"marshal" tool	Salt	Kubernetes	Consul/Vault
F1, F2, F3	Docker	Docker & jar	"marshal" tool	Chef	Native	Chef
G1, G2	Docker	tar	"stackctl" tool	Salt	Native	Salt
H1, H2, H3, H4	Docker	debian	"stackctl" tool	Salt	Native	Salt
I1	None	tar	custom bash + CloudFormation	CloudFormation	Native	S3/KMS

Productivity Inhibitors

- Difficult to change teams
- Microservice maintenance challenges
- Local development was hard
- Creating new microservices was hard
- Too many deployment Jenkins jobs

R&D Productivity

Number of Developers × Developer Efficiency

The campaign for change.

Kube Squad

What are we going to build?

Design Objectives

- Build a fully automated system test pipeline
- Simplify local development
- Simplify deploying services
- Simplify microservice bringup

How did we build it?

Container Standardization Metaphor

Old Way

Ad hoc and Manual



vs.

New Way

Standardized and Automated



Image

microservice
code



microservice
dependencies





Pod

Container

microservice
code



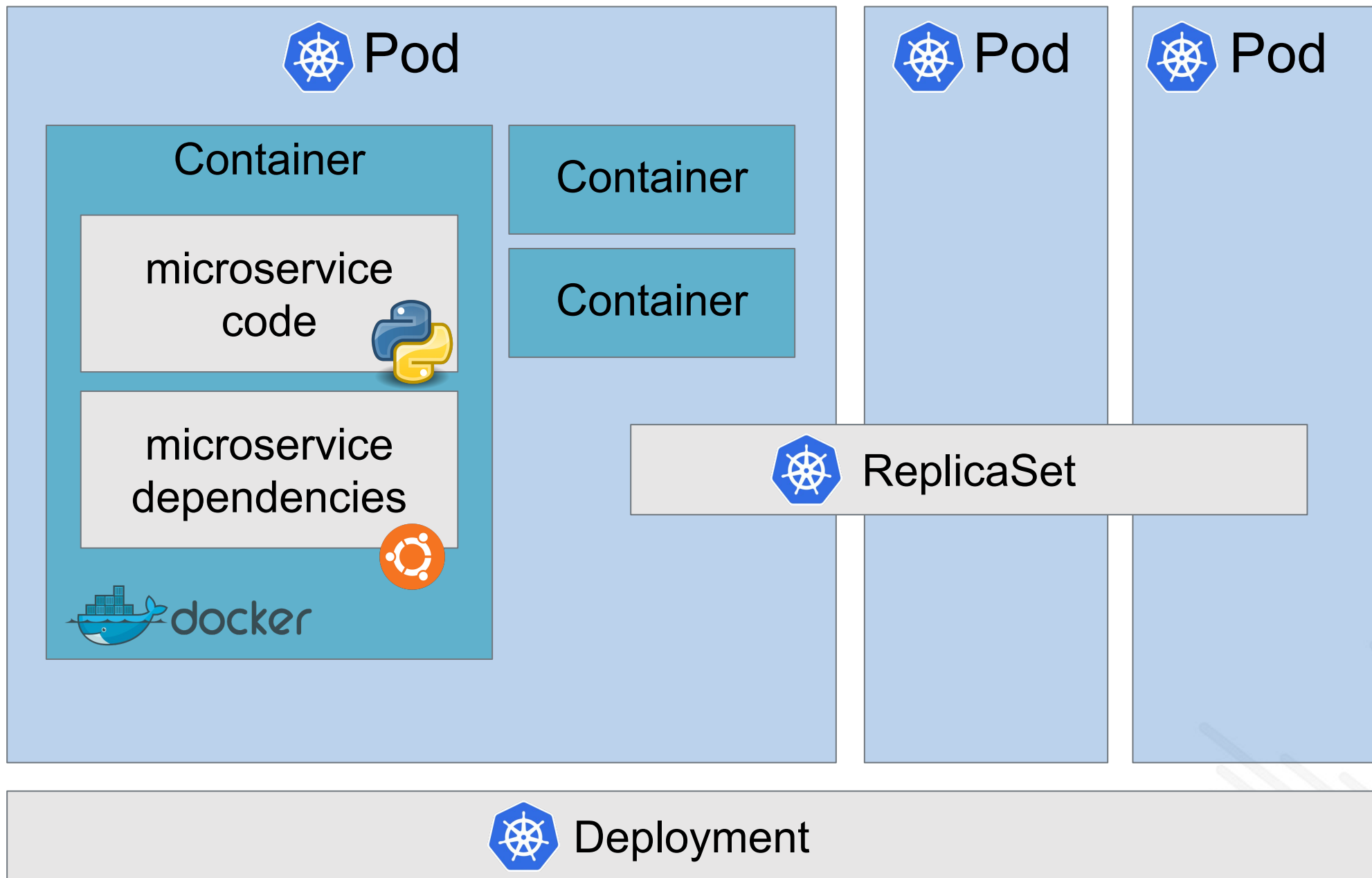
microservice
dependencies



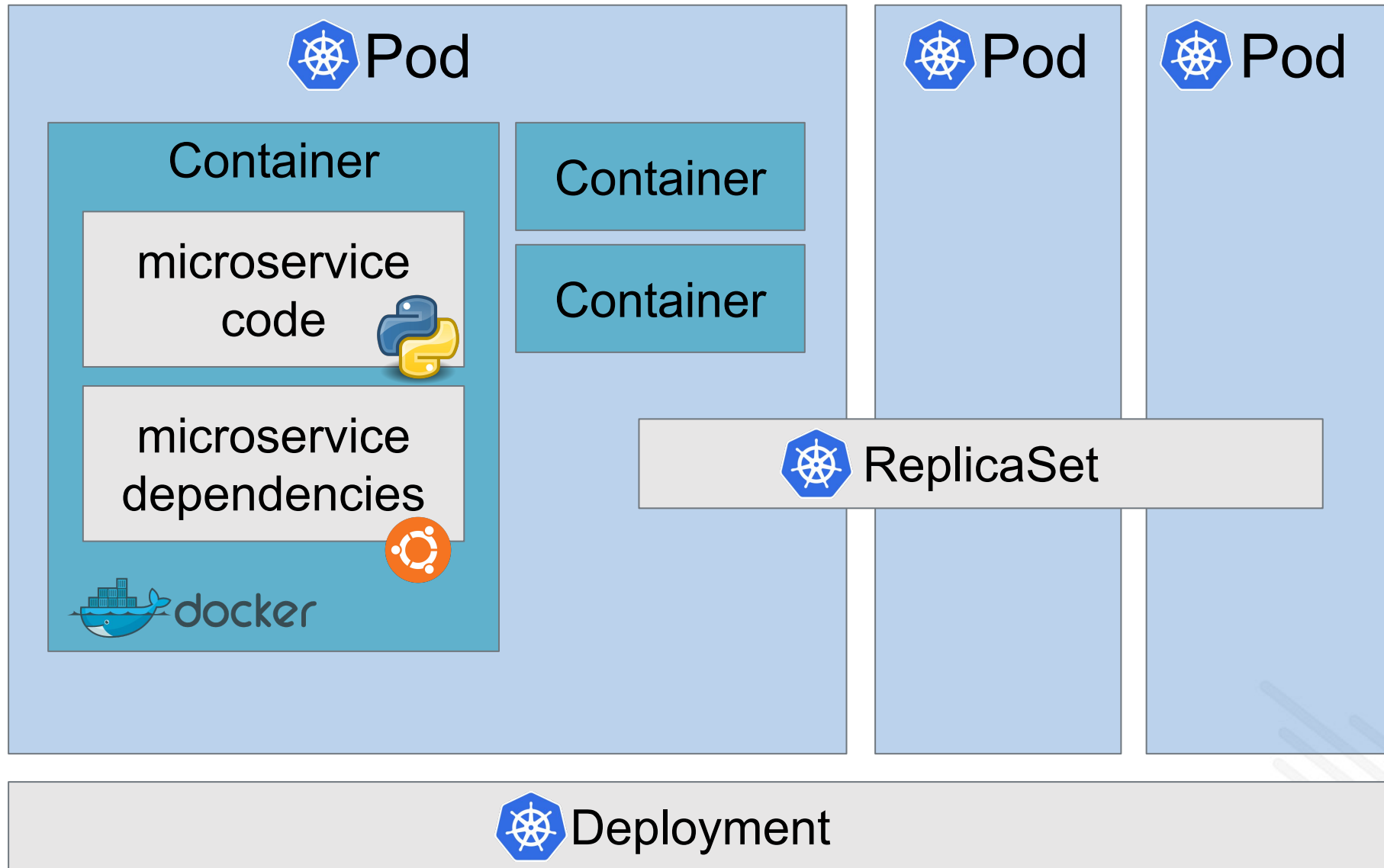
docker

Container

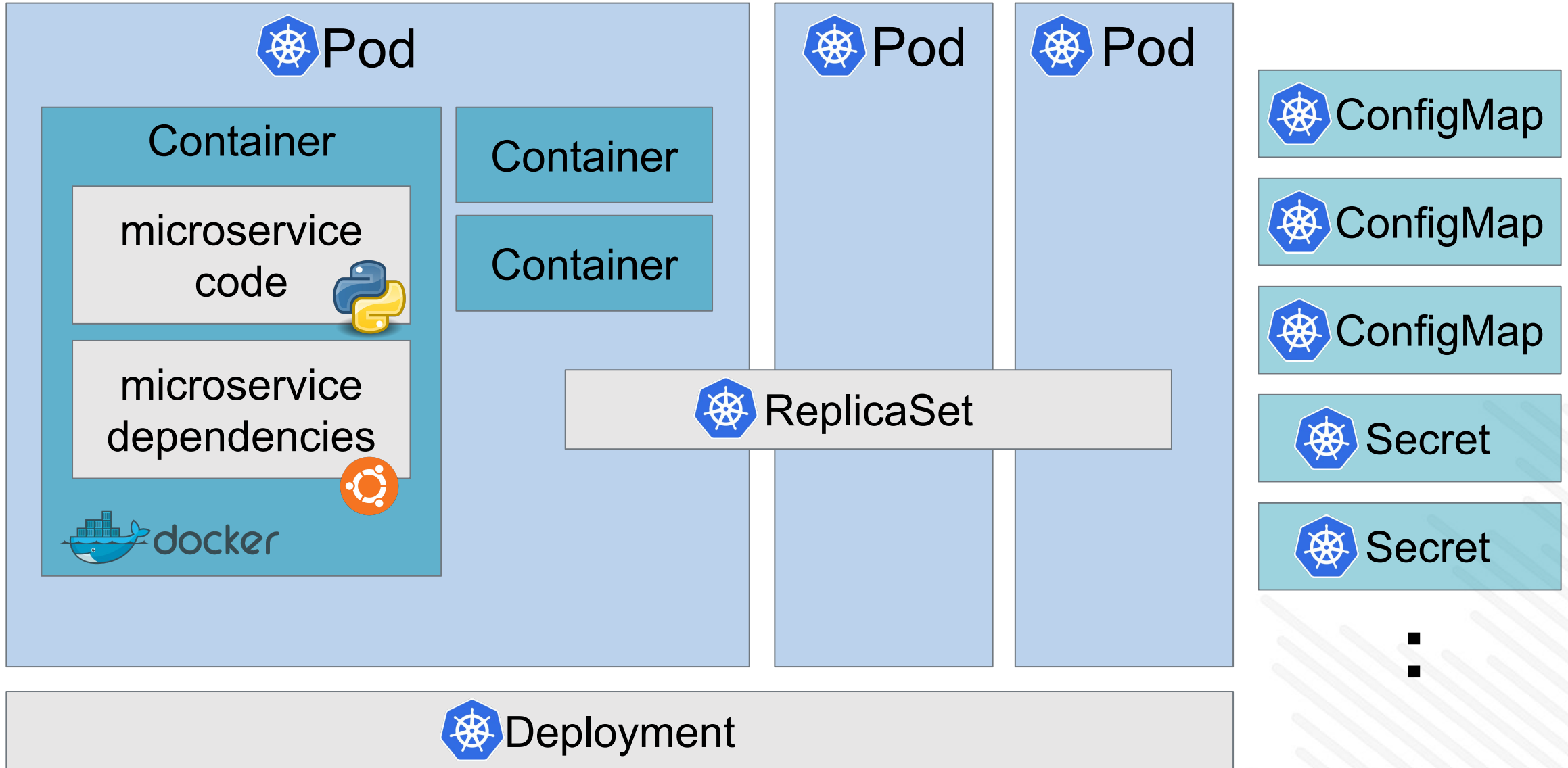
Container

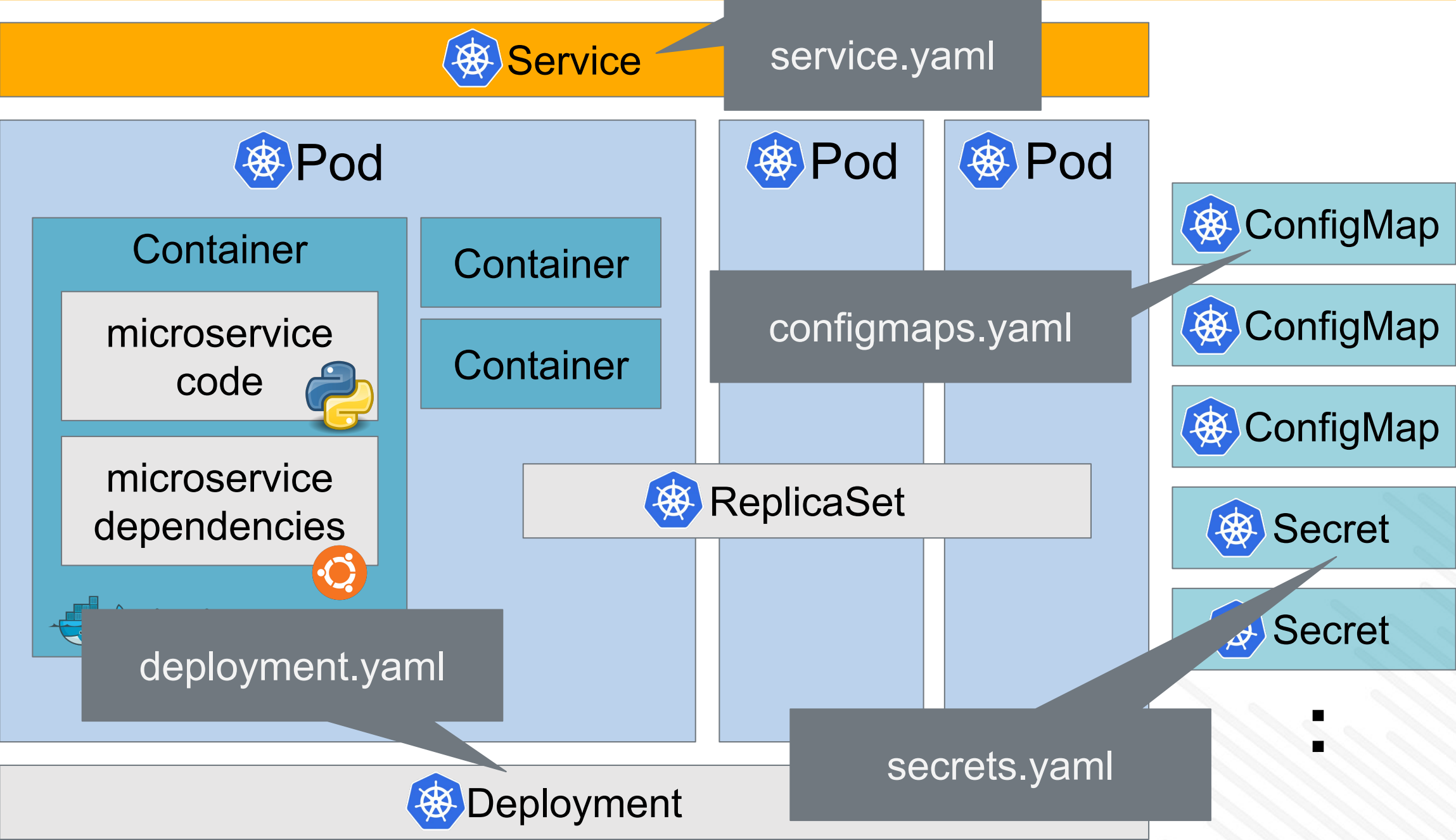


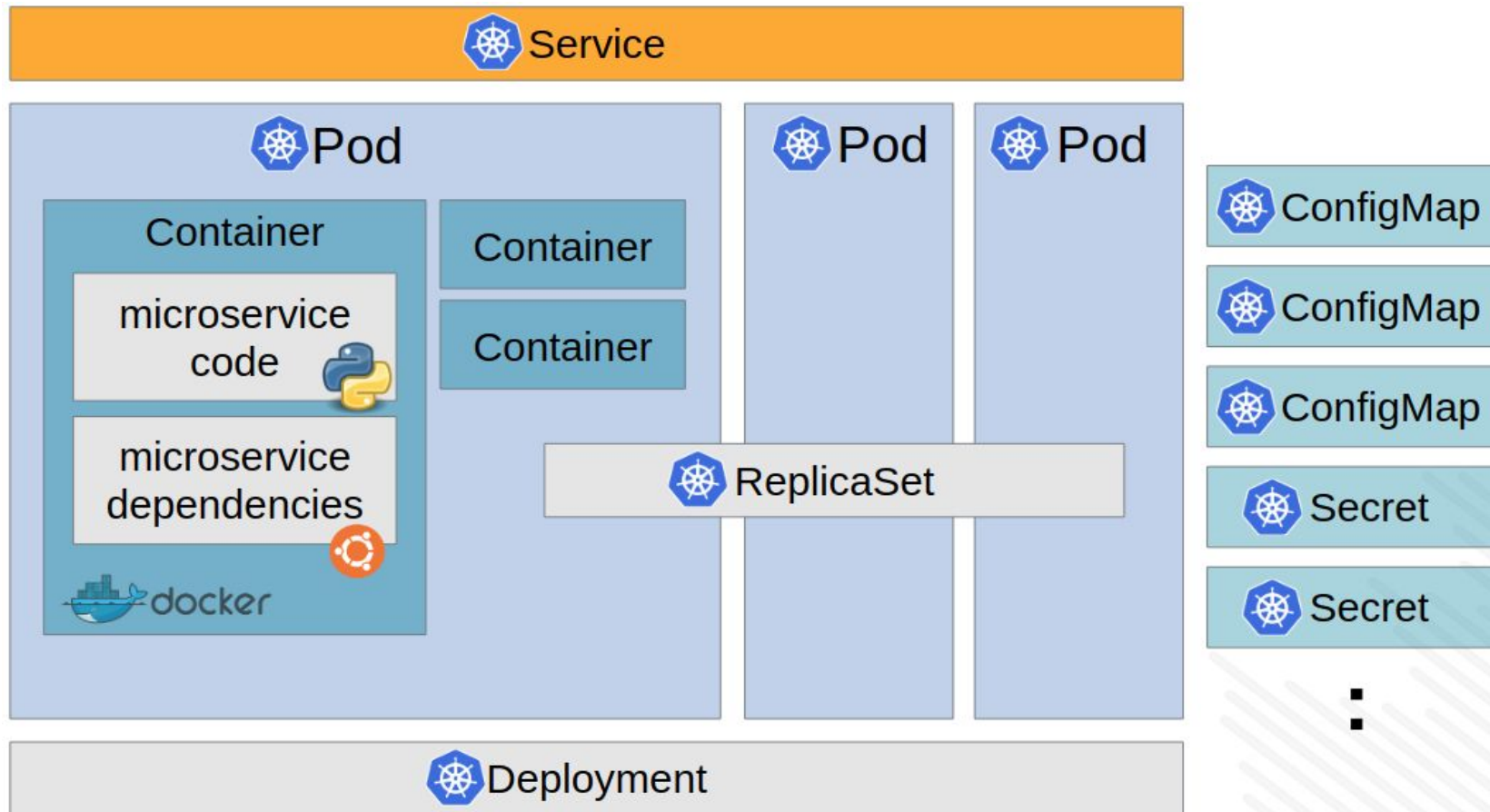
Service



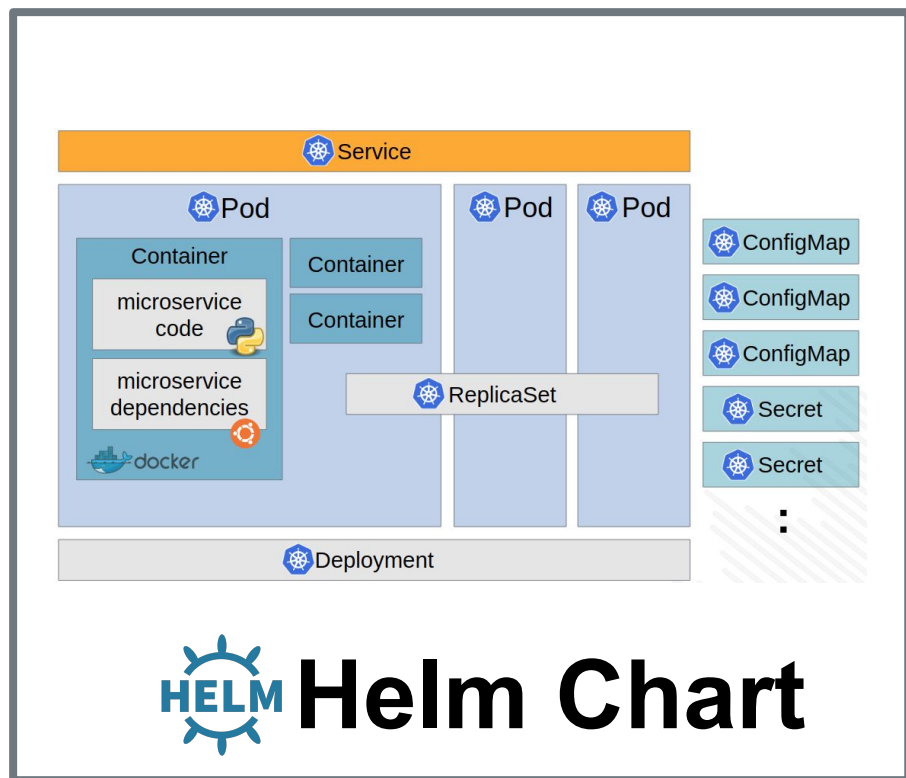
Service



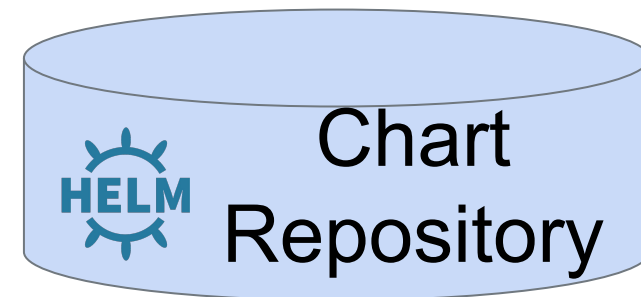




Helm Chart



package



upload



accounts-1.1164.434.tgz

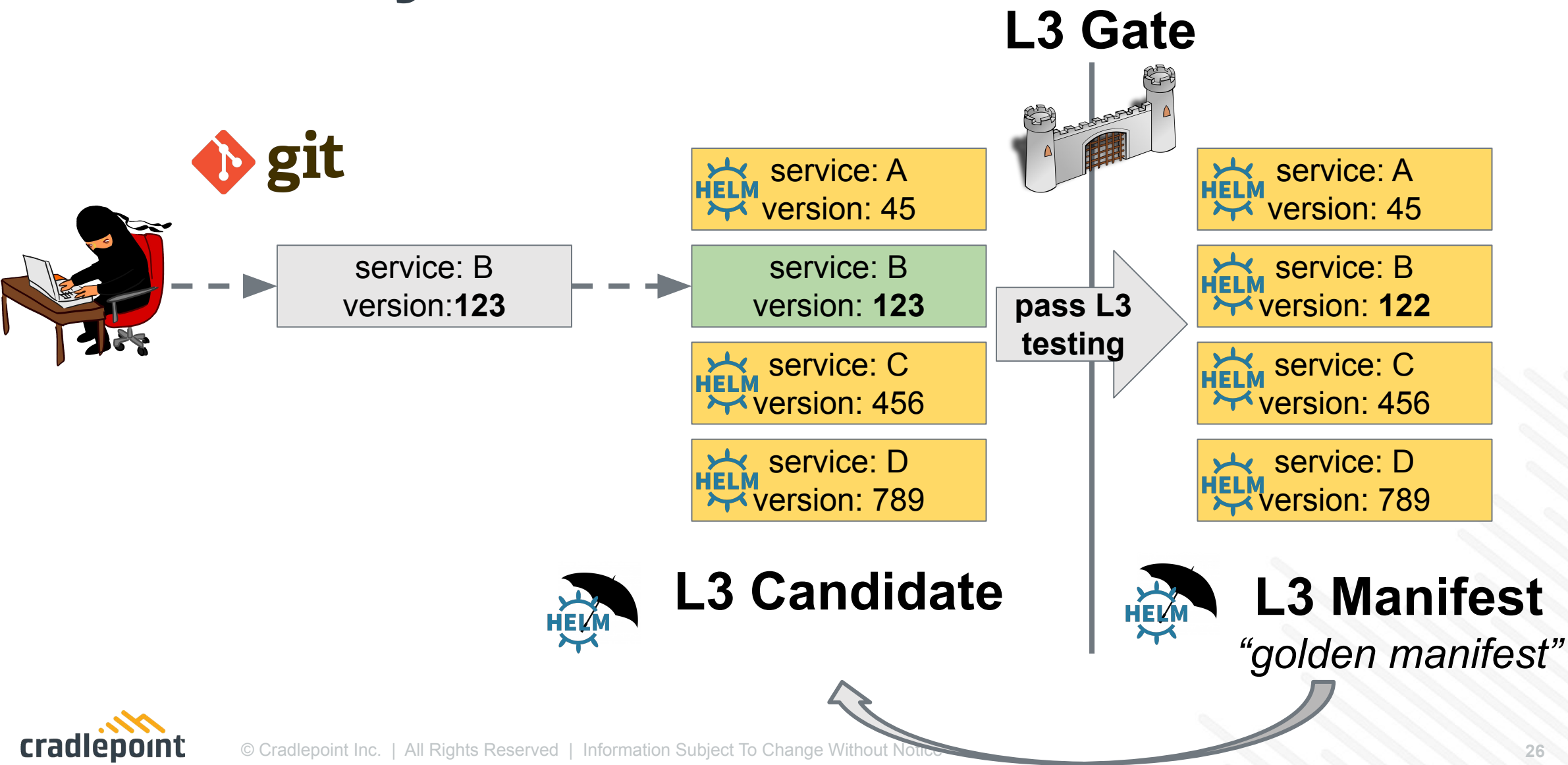
Helm Umbrella Chart “Manifest”



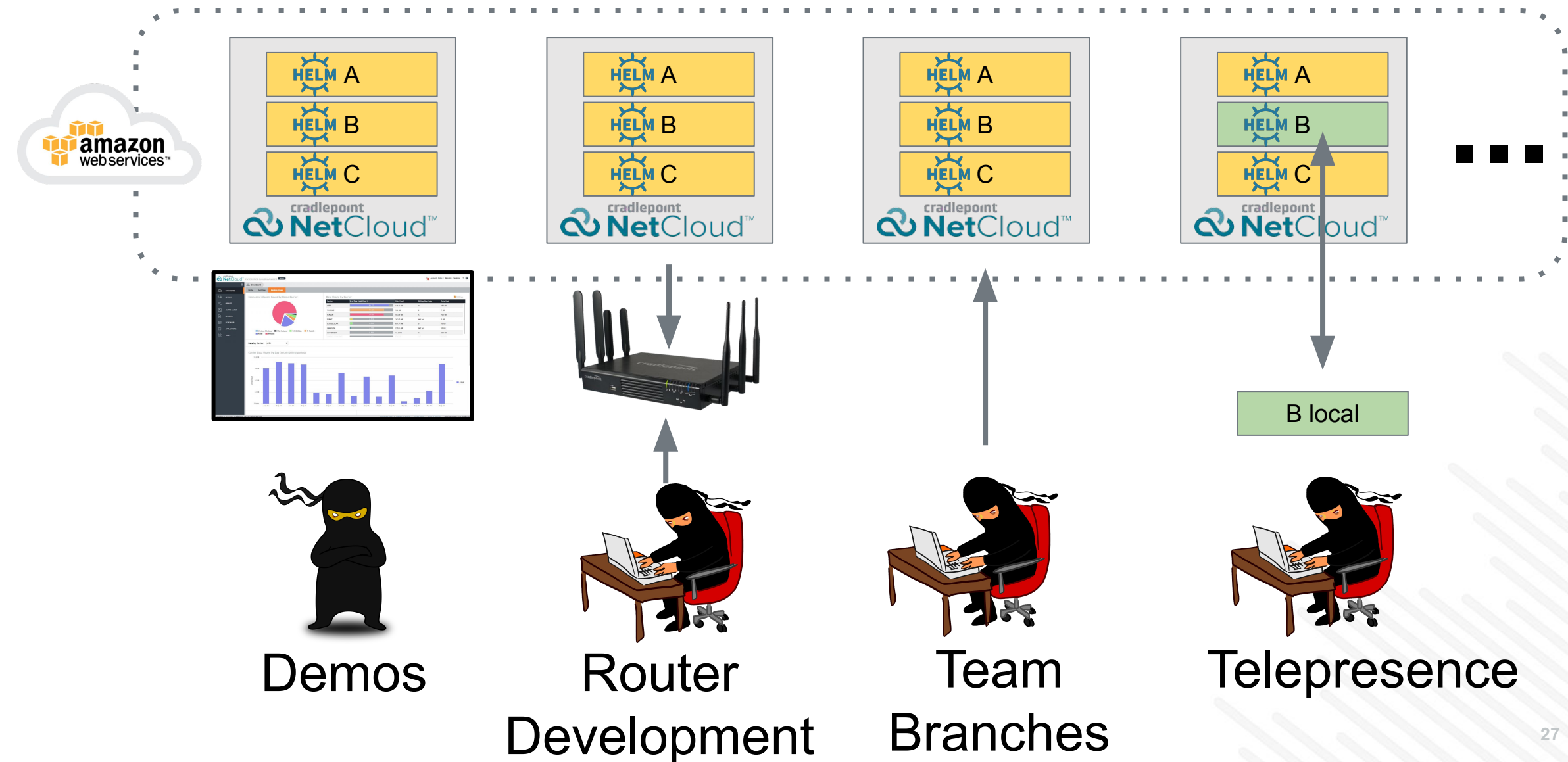
requirements.yaml 798 Bytes

```
1 dependencies:
2   - name: ncm-core
3     version: 1.12345.3807
4     repository: https://example.com/cp-helm-all
5   - name: ncm-data
6     version: 1.12345.3807
7     repository: https://example.com/cp-helm-all
8   - name: accounts-data
9     version: 1.5432.434
10    repository: https://example.com/cp-helm-all
11   - name: accounts-webserver
12     version: 1.5432.434
13     repository: https://example.com/cp-helm-all
14   - name: ncm-streamserver
15     version: 1.456.1185
16     repository: https://example.com/cp-helm-all
17   - name: ncm-cms
18     version: 1.654.80
19     repository: https://example.com/cp-helm-all
20   - name: mailhog-aws
21     version: 0.2.0
22     repository: https://example.com/cp-helm-all
23     condition: global.mailhog.enabled
24   - name: activity-webserver
25     version: 1.987.196
26     repository: https://example.com/cp-helm-all
27   - name: activity-webserver
```

L3 Quality Gate



MyStack: Single-click Developer Stack in AWS



More about MyStacks

- Creation takes about 10 minutes
- Auto-deletes if not used for 7 days
- Default is small but can opt-in for more services
- Developer dashboard for control and custom urls

Simple deployment to any stack

Pipeline upgrade-release

This build requires parameters:

CLUSTER

stage▼
stack to deploy

CHART

accounts-webserver▼
chart to deploy

CHART_VERSION

1.12345.987
chart version ex: 0.1.147+37 or latest

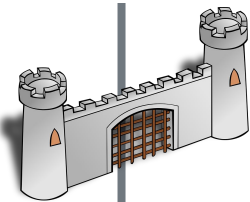
DRY_RUN

☐
(optional) do a helm dry run


Build


Continuous Deployment to Test Stacks


L3 Gate




pass L3
testing

 service: A
version: 45

 service: B
version: 122

 service: C
version: 456

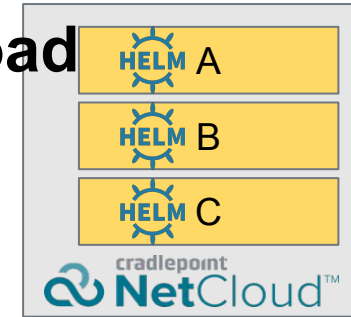
 service: D
version: 789



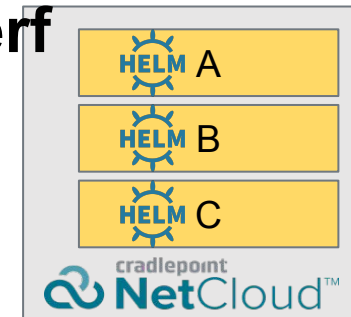
L3 Manifest
“golden manifest”

auto
deployment

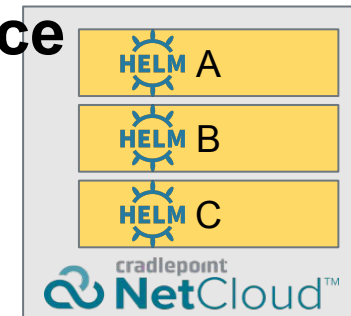
Load



Perf



Salesforce



■ ■ ■

Full chart visibility in every deployment


 Cradlepoint Kubernetes Dashboard

Chart Pipeline

MyStacks

Clusters

Nightly

QA1

QA2

QA4

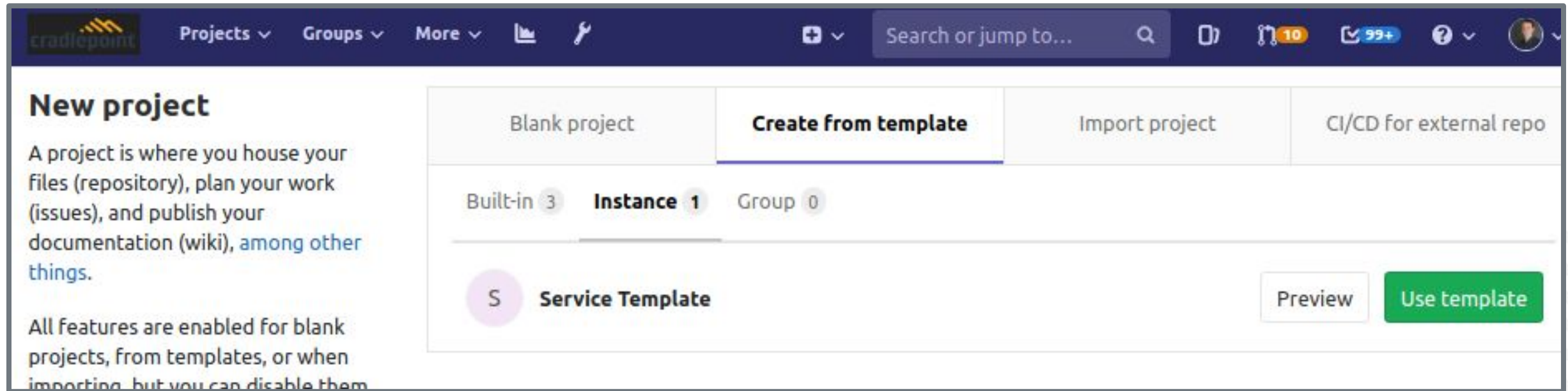
Stage

Prod

Updated 4 minutes ago

CHART NAME	LATEST	L3	NIGHTLY A V5	NIGHTLY B V5	QA1 A V5
accounts-data	1.1166.535	1.1166.535	1.1166.535	1.1166.535	1.1145.2
accounts-webserver	1.1166.535	1.1166.535	1.1166.535	1.1166.535	1.1145.22
alerts-ui	1.130.190	1.128.182			
als-webserver	1.249.196	1.249.196	1.249.196	1.249.196	1.249.196

GitLab Project Template



The screenshot shows the GitLab 'New project' interface. The top navigation bar includes the Cradlepoint logo, 'Projects', 'Groups', 'More', and a search bar. The main content area is titled 'New project' and includes a description: 'A project is where you house your files (repository), plan your work (issues), and publish your documentation (wiki), among other things.' Below this, it states: 'All features are enabled for blank projects, from templates, or when importing, but you can disable them.' The 'Create from template' tab is selected, showing a list of templates. The 'Instance' tab is active, displaying a single template: 'Service Template' with a purple 'S' icon. To the right of the template list are 'Preview' and 'Use template' buttons.

New project

A project is where you house your files (repository), plan your work (issues), and publish your documentation (wiki), among other things.

All features are enabled for blank projects, from templates, or when importing, but you can disable them.

Blank project **Create from template** Import project CI/CD for external repo

Built-in 3 **Instance 1** Group 0

S **Service Template** Preview **Use template**

Achieved Design Objectives

- ☑ Build a fully automated system test pipeline - **L3**
- ☑ Simplify local development - **MyStacks**
- ☑ Simplify deploying services - **Auto Promotion**
- ☑ Simplify microservice bringup - **~ 1 hour**

Productive Engineers are Happy Engineers



Lessons Learned

- Build a Kubernetes cluster pipeline
- Helm 2 has some warts
- L3 is complex and takes investment
- Microservices are still hard

Future

- Complete last few charts on production
- Service Mesh
- Helm 3
- kops to EKS
- Canary Deployments

Thank You!

- Come visit the Cradlepoint table in the lobby!
- Twitter @BoiseMatt
- R&D Blog @ <https://cradlepoint.com/blog>