Operating a Global Cloud Platform

Josh Michielsen
@jmickey_
Using Traefik as an ingress controller for public and private ingress for our Kubernetes clusters.

How we manage identity and authentication across multiple clusters.

Helm simplifies the packaging and deployment of applications running on Kubernetes.

Managing out-of-cluster infrastructure with Terraform, and cultivating an inner-source community around it.
Landscape Overview
Overwhelmed? Please see the CNCF Trail Map. That and the Interactive landscape are at l.cncf.io.
Everything Else

Service Mesh

Kubernetes Landscape Word Cloud
Platform Overview
Global Cloud Platform

- Clusters in 4 Regions
- 11 Markets
- 180m+ Monthly Pageviews
- 23/34 Publications Migrated
03

Logging
Fluentd is an open source data collector for unified logging. It provides an easy way to retrieve, process, format, and forward application logs.
Fluentd at Condé

→ Application developers configure their apps to log to stdout.
→ All development teams must adhere to our structured logging standard.
→ Fluentd is deployed as a Kubernetes DaemonSet within its own namespace.
→ Fluentd is configured with access to the local node logs, and the Kubernetes log volume.
→ Logs are processed with additional metadata (e.g. namespace, labels, env, region).
→ Logs are then forwarded to AWS ElasticSearch via a cluster local ES proxy.
The format for the log line. In this case Kubernetes.

Interval between buffer flushing.

Location of the log file in the node file system.

Store the last position read within the log file.

Tag the log blog with the Kubernetes service.
Monitoring
Datadog is a cloud-based metrics and monitoring service. Commonly used for monitoring and alerting on infrastructure, as well as Application Performance Monitoring (APM).
Datadog at Condé

→ Deployed via Helm.
→ Two DaemonSets. One for master nodes, another for workers.
→ Kubernetes PriorityClass on master agents to protect from descheduling.
→ As with all monitoring and alerting, experience is heavily dependant on the implementation.
→ Very little configuration required. Great for quickly getting started.
Learnings

→ Can quickly become expensive as development teams increase the number of custom metrics.
→ Fairly steep learning curve for querying language and formulas.
→ Documentation could be better.
→ Investigation of Prometheus and Thanos for multi-cluster aggregation on the roadmap.
Ingress
A modern HTTP reverse proxy and load balancer that makes deploying microservices easy. Traefik integrates with your existing infrastructure components and configures itself “automatically and dynamically”.
Internet

- api.example.com
- example.com/web
- docs.example.com
- private.example.com

Private

Orchestrator

- API
- Web
- Docs
- Private (3 instances)
Traefik at Condé

→ Each development team has a namespace.
→ Each namespace has a public ingress, and a private ingress.
→ Certificates are configured on AWS ELBs via AWS ACM.
→ Ingress rules are managed via an ingress configuration block within the Helm chart.
→ Enables developers to manage their own application ingress rules. Including allow and block lists.
ingress:
  enabled: true
  traefik: traefik-namespace.private.r53.dns
  type: external
  annotations:
    traefik.ingress.kubernetes.io/frontend-entry-points: http,https
    traefik.ingress.kubernetes.io/redirect-entry-point: https
  rules:
    - host: api.myapp.private.r53.dns
      path: /
Authentication
Federated OpenID Connector (OIDC) by CoreOS. It acts as a portal that defers authentication to third-party identity providers (IDP) such as Active Directory, SAML, or cloud providers like GitHub and Google.
Auth at Condé

→ GitHub is our IDP, and permissions are managed via GitHub “teams” and Kubernetes RBAC.
→ Okta adopted since the launch of the platform. Migration from GitHub to Okta planned.
→ Custom developer authentication portal that provides a simplified workflow for authenticating with clusters.
→ Service account token are provided within CI/CD pipelines - not visible to developers and rotated periodically.
Kubernetes Auth

Welcome to our Kubernetes cluster 🤝
Have a nice day!

SIGN IN

Congratulations🎉

You have successfully authenticated with your authentication provider to enable access to our Kubernetes cluster.

Run the following command locally, to ensure `kubectl` has the appropriate configuration for this environment.

COPY COMMAND

```
kubectl config set-cluster prod_eu-central-1 --server=
kubectl config set-context prod_eu-central-1 --cluster=prod_eu-central-1 --user=github#
kubectl config set-credentials github#mickey.dev-prod_eu-central-1 --auth-provider=oidc
kubectl config use-context prod_eu-central-1
```

If this is your first time connecting to this environment, use the following to setup your default namespace.

```
kubectl config set-context $(kubectl config current-context) --namespace=<a namespace>
```

To confirm everything is working as expected, and that you can access this cluster, please test by running the command `kubectl get pods`.

▶ I’m feeling 😊

https://github.com/conde-nast-international/kubernetes-auth

@jmickey_
Learnings

→ Inconsistent permissions management between GitHub and Okta. Not a massive issue, but does have a small overhead.
→ Authentication is not federated across clusters. Devs need to authenticate to each cluster they want to query.
Application Delivery
A Kubernetes package manager that simplifies the packaging, configuration, and deployment of applications and services onto Kubernetes clusters
Helm Basics

Provides a templating language that can be used to generate standard resource configurations. Charts can be provided a set of override values.

Helm charts can have dependencies, allowing you to modularise your Helm configurations.

When executed, Helm:

→ Replaces the values in the configuration
→ Builds the resource definitions
→ Deploys them to Kubernetes, and keeps track of all those associated resources
→ All while versioning them as a set (A.K.A a “release”)

$ helm create myapp
$ cat myapp/templates/deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "myapp.fullname" . }}
labels:
{{ include "myapp.labels" . | indent 4 }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app.kubernetes.io/name:
        {{ include "myapp.name" . }}
      app.kubernetes.io/instance:
        {{ .Release.Name }}

...
Helm at Condé

→ Single base helm chart used across all development teams.
→ YAML file to provide values for each environment, stored in the application repo.
→ Conditionals on dependencies means developers can choose the features they want to use by simply specifying the config for that feature.
→ We set non-negotiable Helm configuration items that must be included (e.g. Limits).
→ Deployed to Kubernetes from CircleCI.
Dependency: Ingress
Condition: ingress.enabled

Dependency: HPA
Condition: hpa.enabled

Dependency: Service
Condition: service.enabled

Base

Helm
Chart

myapp/prod.yaml

name: myapp
replicas: 3

ingress:
  enabled: true
...

service:
  enabled: true
...
Supporting Infrastructure
Terraform provides a declarative language for provisioning, changing, and managing infrastructure for a wide range of tools and services.
Terraform at Condé

→ Terraform code is declared once and reused across environments and regions through variable injection.
→ Continuous delivery pipelines are configured so that devs can update infrastructure without waiting for platform teams to apply changes.
→ Central modules repo that anyone can contribute to.
→ Devs are encouraged to write their own infrastructure code, with PRs being approved by platform.
Terraform at Condé

$ cd prod/eu-central-1/route53

$ terraform plan -var-file=terraform.tfvars -out=prod-eu-central-1-route53.plan

$ terraform apply prod-eu-central-1-route53.plan

$ cd prod/eu-central-1/route53

$ terraform plan -var-file=terraform.tfvars -out=prod-eu-central-1-route53.plan

$ terraform apply prod-eu-central-1-route53.plan
Learnings

→ We were overzealous with modules.
→ The automation of planning and applying terraform is mostly held together by bash scripts. These can be difficult to maintain.
→ IAM permissions for automation CI/CD keys took a little while to get right.
→ Plans are reviewed manually and manual approval is required in CD before apply can happen. Investigating ways to run checks against plans so that this can be automated a bit more.
The Future
Prometheus → The introduction of tools like Thanos and Cortex have made managing Prometheus across multiple clusters, envs, and even namespaces much easier.

Weaveworks Flux → GitOps for Kubernetes. Git becomes the single source of truth, and Flux executes automatic remediation when drift occurs.

Service Mesh → mTLS throughout the cluster, retries, service discovery, load balancing, auth(n/z).
Thanks for Listening
Please Rate this Session
We’re Hiring! Come Chat

CONDÉ NAST

mickey.dev    @jmickey_
j@mickey.dev   jmichielsen
jmickey