Forced Evolution: Shopify's Journey to Kubernetes
Shopify

3000+ Employees

$26B processed ‘17

600k+ merchants

80+k Peak RPS
Running services.. everywhere

- DCs
  - Chef+docker
- AWS PCI
  - Chef
- AWS
  - Chef+???
- Heroku
Service Tiers

- Tier 1:
  - More mature in SDLC
  - Regional redundancy, incident response drilling

- Tier 2:
  - Greater business importance
  - Pager rotation, automated critical alerting

- Tier 3:
  - Higher SLO
  - CI, Pingdom, backups, logging

- Tier 4:
  - Earlier in SDLC
  - Fewer requirements to encourage rapid prototyping
Not scalable
Things that won’t scale

- Manual / Artisanal processes
- Slow things/processes that make people wait
- Rusty knobs that don’t work when needed
- Wobbly things that don’t work first-time, every-time
Things that will scale

- Tested infrastructure
- Automation that works as expected, every time
- Give devs ability to self-serve with safety
- Train people to be experts in the systems they operate
Building a PaaS
“One Ring to rule them all, One Ring to find them, One Ring to bring them all and in the darkness bind them”

- The Lord of the Rings
Three principles
Paved road
Hide complexity
Self serve
Why Kubernetes?

- Best traction of the open source projects
- Platform agnostic
- One of the most extendable solutions
- Written in Go
- Offered as a service in Google Cloud
Building blocks of running an application

- How to specify your apps runtime
- How to build your app
- How to deploy your app
- How to set up your dependencies
Creating application environment

Services DB
- Web UI for developers
- Application catalog
- Generation of Kubernetes manifests
- Configures builds and CI

Groundcontrol
- Go app living on clusters
- Creates k8s namespace
- Creates encryption keys
- Service accounts
Welcome to Shopify Cloud!

This PR contains the Kubernetes files and Shipit script you will need to deploy this app.

Please review this PR before merging it:

- Review the Kubernetes templates to familiarize yourself with the kinds of configuration stored here.
- Make sure k8s-shell.yaml contains everyone needing Kube Shell access to your containers.
- We've configured the public key in this config/secrets.env.toCharArray() on our cloud servers. If you have a previous public key in place, you will need to re-encrypt your secrets.
- If you need to add environment variable secrets (e.g., you are migrating from Heroku), you can add them to the "environment" key of config/secrets.env.toCharArray(). Don't forget to run envars encrypt before committing!

Questions about this PR? Read the docs or hop in the #services Slack channel. We're happy to help!
Buildkite + PIPA

- Buildkite acts as coordinator for Pipa
- Pipa agent builds Docker images
- Herokuish, Dockerfile, or custom build pipelines
Builder Stats

6,000 average builds per weekday

450,000 images in GCR
- Pass/fail results on deploys
- Pre-deploy for ConfigMap/Secrets
- Protecting namespaces
- Pluggable
Cloudbuddies

- Create DNS records
- Fetch SSL certificates
- Create buckets, databases, services etc
- Set user editable quotas
- Set security rules
- Delete bad nodes
**Namebuddy flow**

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: web-nginx
  annotations:
    kubernetes.io/ingress.class: nginx
    kubernetes.io/tls-acme: "true"
labels:
  name: web
  app: cumulus-cat
  env: production
spec:
  tls:
    - hosts:
      - cumulus-cat.shopifycloud.com
```

Listen to Ingress objects

Create/delete/update DNS record

Read objects hosts field and IP address

create delete update

create delete update

K8s API

DNS provider API

```
;:docs.shopifycloud.com. IN A
;; ANSWER SECTION:
docs.shopifycloud.com. 59 IN CNAME
dcloud-docs-production-nginx-tier3.shopifycloud.com. 59 IN A
35.185.75.173
```
Crash course to buddies
Extending k8s

- API's are well documented (if not super stable)
- Client libraries are high quality (at least on client-go)
- We can both extend functionality of current concepts (deployments, endpoints etc) but also create our own (CRDs)
- Distributed systems primitives (leader election, latches ...)
- These apps are be pure Go so they are unit testable, running and deployed as normal apps etc.
Kubernetes Controllers

An active state reconciliation process

- Watch desired and current state
- Try to mutate desired to current

```plaintext
for {
    desired := getDesiredState()
    current := getCurrentState()
    if desired != current {
        reconc(desired, current);
    }
}
```
Workflow is always the same

- Authenticate to the cluster
- Create a watcher for events of specified type
- Implement functions to handle ADD/DELETE/UPDATE
- Profit!
Custom Resource Definitions

- Extend native k8s objects with your own abstractions
- Eg. Memcache, Redis, Mail, MyFancyThingy
- Used by your own controllers to consume configuration params and doing something based on it
- Just like normal k8s resources like Deployment or Service
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
labels:
  app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
spec:
containers:
- name: nginx
  image: nginx:1.7.9
  ports:
  - containerPort: 80
apiVersion: stable.shopify.io/v1
kind: Elasticsearch
metadata:
  name: <%= @app %>
  labels:
    app: <%= @app %>
    environment: <%= @env %>
    component: elasticsearch
spec:
  elastic-search-version: '6'
  zones:
    - us-east1-b
    - us-east1-c
    - us-east1-d
  elasticsearch-spec:
    reindex.remote.whitelist: 10.*.*.*:9200
  node-specs:
    replicas: 3
    cpu-limit: "1"
    mem-limit: 2G
    data-volume-size: 10Gi
  snapshot:
    bucket-name: shopify-<%= @app %>-%<%= @env[0..3]%>-es-snapshots
Supporting users
### Managed workloads

<table>
<thead>
<tr>
<th>Name</th>
<th>Pods Status</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>backgroundqueue-high-priority</td>
<td>3/3 pods ready</td>
<td>7 months ago</td>
</tr>
<tr>
<td>backgroundqueue-long-running</td>
<td>5/5 pods ready</td>
<td>7 months ago</td>
</tr>
<tr>
<td>backgroundqueue-low-priority</td>
<td>25/25 pods ready</td>
<td>7 months ago</td>
</tr>
<tr>
<td>cloudsql-07b3a605-b9b2-11e7-bb59-42010af00076</td>
<td>2/2 pods ready</td>
<td>8 months ago</td>
</tr>
<tr>
<td>memcached-apo-reviews-production</td>
<td>1/1 pods ready</td>
<td>8 months ago</td>
</tr>
<tr>
<td>redis-e08649e1-b9b2-11e7-bb59-42010af00076</td>
<td>1/1 pods ready</td>
<td>8 months ago</td>
</tr>
<tr>
<td>scheduler</td>
<td>1/1 pods ready</td>
<td>7 months ago</td>
</tr>
<tr>
<td>vividcortex-07b3a605-b9b2-11e7-bb59-42010af00076</td>
<td>1/1 pods ready</td>
<td>8 months ago</td>
</tr>
<tr>
<td>web</td>
<td>12/12 pods ready</td>
<td>7 months ago</td>
</tr>
</tbody>
</table>
Overview

The CloudPlatform runtime is a unified deployment stack for Shopify applications backed by Kubernetes that strives to provide an out-of-the-box experience when possible (see Project Brief).

To use this documentation:

- Browse the categories in the sidebar or enter key terms in the search bar at the top right.
- Read the introduction for a primer if you're just getting started.
- Check out the glossary for clarification on any terms. You may also mouse over terms with a hatched underline to see a quick description.
- If you have read through the docs and still need help (or have an emergency) check out the guidelines on how to get help

Note

This documentation is intended to help make CloudPlatform self-service, without your needing to know too much about how the platform itself works. For knowledge sharing, and to better understand the platform, however, details about how things work are often provided to demystify some of the magic.
Currently on call for Cloud-Help:
Cloud help (primary): @jenna (jenna.black@shopify.com) Until: 2018-06-14T21:00:00Z UTC
Cloud help (secondary): @stefan.budeanu (stefan.budeanu@shopify.com) Until: 2018-06-14T21:00:00Z UTC

Report card
"The turn around time to getting an app running on cloud platform is unreal, you folks have really nailed it."
Challenges for developers

- How does my builds/deploys/everything work?
- How do I scale?
- How do I debug?
- Is this worth it?
Challenges for SREs

- Giving up control over underlying infrastructure
- Container-only world and new tooling
- Customising the one platform to fit all needs
- Constant pressure to migrate apps
- Learning
Takeaways for building your own PaaS

- Target hitting eg. 80% of use cases
- Create patterns and hide complexity (but don’t restrict)
- Educate
- Get people excited
- Be conscious of vendor lock in
Future

• Polishing our tooling
• Making sure our platform keeps scaling and stable
• Optimising cost
• Multi cloud
• Service mesh
Thanks!
• github.com/Shopify/kubernetes-deploy
• github.com/Shopify/kubeaudit
• github.com/Shopify/shipit-engine