BUOYANT

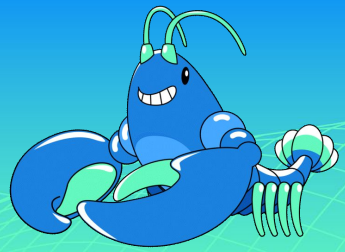
Creators of LINKERD



Linkerd 2.13 in Production 101

Flynn, Technical Evangelist for Linkerd

flynn@buoyant.io | @flynn on slack.linkerd.io



Why are there so many slides??



- For many SMAs, we have few slides and a lot of workshop.
- This is not one of those SMAs. 😂
- There are a lot of slides, with a lot of words, because we want this to be a reasonable reference for reading later.

BUOYANT
Creators of LINKERD

What do you need for the workshop?



- A Kubernetes cluster **with at least three Nodes**. I'll be using `civo`, but the type of cluster doesn't much matter.
- To follow along, you'll also need `kubectl`, `bat`, `step`, and the `linkerd CLI`.
- Check out the workshop source at

  https://github.com/BuoyantIO/service-mesh-academy/tree/main/linkerd-in-production

BUOYANT
Creators of LINKERD

What do you need for the workshop?



- Clone the workshop repo and **MAKE SURE YOU'RE USING AN EMPTY CLUSTER.**

```
cd linkerd-in-production
bash demosh/check-requirements.sh
```

- Even better, use `demosh`: it'll check everything for you.

```
pip install demosh
demosh README.md
```

BUOYANT
Creators of LINKERD

What's on the agenda?



- ➔ **Taking Linkerd to Production**
  - Basic checklist
  - Monitoring
  - Debugging
  - Upgrading/rolling back
  - Gotchas!
- ➔ **Workshop!**

BUOYANT
Creators of LINKERD

What's **NOT** on the agenda?



- **This can't be a deep dive for everything.**
- We 're not going to cover:
  - All the details of certificate management
  - Ingress
  - Policy, traffic management, etc.
- **These all have their own Service Mesh Academy episodes,** which we'll link to.

BUOYANT
Creators of LINKERD

What do we assume you know?



- We assume you're basically comfortable with Kubernetes.
- We assume you're at least familiar with `kubectl` and `linkerd`.
- **Unlike many SMAs, this is not a total-beginner course.**
- Don't panic: we'll be linking out to a **lot** of resources.

BUOYANT
Creators of LINKERD

Demos vs Production

Setting the stage for the rest of this workshop...

DEMO:

- Often a local cluster
- Usually ephemeral (hours, maybe days)
- Crashes aren't important
- No particular security needs
- Hand edits are often the norm
- Expiring certificates annoy the demo owner

PRODUCTION:

- Usually a cloud-provider cluster
- Usually long-lived (though perhaps shouldn't be?)
- Uptime is critical
- Often need to adhere to corporate security
- Controlled configuration is critical
- Expiring certificates cause downtime

Production Checklist for Linkerd

Mostly from Buoyant's Linkerd Production Runbook:

https://buoyant.io/linkerd-runbook



Productionalization checklist

1. Think carefully about certificates and your CNI
   - We strongly recommend `cert-manager`
   - We generally recommend using the init container instead of the CNI plugin.
2. Put your configuration in version control
   - Consider GitOps, but **definitely** use version control!!
   - We recommend Helm, too
3. Run in high-availability mode (HA) with your own image registry
4. Lock down access wherever you can
5. Set up monitoring and alerting
6. Understand how to install and upgrade Linkerd
7. Understand how to debug Linkerd

(Spoiler alert / shameless plug: check out Buoyant Cloud, too.)

BUOYANT
Creators of LINKERD

1a. Think carefully about certificates

- Expired certificates are **the** most common reason for production Linkerd problems.
- You really, really do not want this to happen to you.
- Let your trust anchor certificate live in an off-cluster secrets manager
  - Don't ever have the trust anchor secret key into the cluster
- Use `cert-manager` to rotate the identity issue certificate frequently
  - e.g. every 48 hours

Cloud Native Live on certificate management with Linkerd:
`https://www.youtube.com/watch?v=TC665X-uzcQ`

(Spoiler alert: this'll also be next month's SMA. 🙂)



1b. Think carefully about your CNI

- ➔ Linkerd *must* tweak the network to make mesh routing work, but it can do this either with an init container or with the Linkerd CNI plugin.
- ➔ We generally recommend the init container, unless you can't give Linkerd the `NET_ADMIN` and `NET_RAW` capabilities.
- ➔ This is simply because the CNI, in general, offers more opportunities for something to fail.

SMA on the Linkerd startup process:

`https://buoyant.io/service-mesh-academy/what-really-happens-at-startup`



2. Put your configuration in version control

- ➔ Making configuration edits you then forget about is the **second** most common reason for production Linkerd problems.
- ➔ We think GitOps is a great answer here.
- ➔ (You can get a long way by just using Helm and `git commit`, though.)

SMA on GitOps and Linkerd:

`https://buoyant.io/service-mesh-academy/real-world-gitops-with-flagger-and-linkerd`



Aside: why Helm?

- The YAML that Linkerd publishes is **not** what you want to be running for production; you're going to need to make changes.
  - don't use our Prometheus, do use HA mode, etc.
- Maintaining your own `values.yaml` file is simpler than maintaining a bunch of `kubectl` commands.
  - and also easier to keep in version control!

SMA on GitOps and Linkerd:

`https://buoyant.io/service-mesh-academy/real-world-gitops-with-flagger-and-linkerd`



2. Put your configuration in version control

- ➔ The critical bit here is that you when you're in production, you need to **know** what's in your cluster.
- ➔ This means that you **must not** have people making *ad hoc* changes to your production cluster.
  - ◆ (says the voice of long and painful experience)

SMA on GitOps and Linkerd:

`https://buoyant.io/service-mesh-academy/real-world-gitops-with-flagger-and-linkerd`



3a. Run in High Availability Mode

- Three replicas of control plane components, on separate Nodes
  - No single points of failure
- Won't allow Pods to start until the proxy-injector is up
  - Guarantees mTLS
- Sets resource requests on control plane components
  - Helps Kubernetes' scheduling algorithms

Linkerd HA mode documentation:
https://linkerd.io/2.13/features/ha/



3a. Run in High Availability Mode

Gotchas:

- Requires at least three Nodes!
  - Node anti-affinity prevents two replicas on the same Node
- Requires the `config.linkerd.io/admission-webhooks=disabled` label on the `kube-system` namespace
  - Don't let Linkerd inject proxies in `kube-system`

Linkerd HA mode documentation:
https://linkerd.io/2.13/features/ha/



3b. Use your own image registry

- ➔ Linkerd publishes images to a single registry (GHCR)
- ➔ Works great until a Pod needs to start and GHCR is down or unreachable
- ➔ Running your own image registry keeps this under your control

Using a Private Docker Registry with Linkerd:

`https://linkerd.io/2.13/tasks/using-a-private-docker-repository/`



4. Lock down access wherever you can

- Linkerd has been adding more and more tools for access control
- Least privilege: close off access that isn't necessary
  - e.g. does the microservice serving the UI's HTML really need to able to check the user's bank balance? probably not!
- The simplest approach here is to lock down whole namespaces, then add routes as needed.
  - Or `linkerd viz profile --tap` can write a ServiceProfile based on traffic as a starting point

SMA on Linkerd 2.12+ policy:

`https://buoyant.io/service-mesh-academy/a-deep-dive-into-route-based-policy`



5. Set up monitoring and alerting

- **Do not use the `linkerd viz` Prometheus in production.**
  - The best home for your metrics is off-cluster
  - Buoyant Cloud, external Prometheus, Datadog, etc
- Do set up alerts on control plane and data plane components.
  - Control plane components should always* show 100% success
  - Latency and resource usage need to be determined empirically
- Don't forget to alert on TLS certificate expirations!

** basically always, anyway*

Using an external Prometheus with Linkerd:
`https://linkerd.io/2.13/tasks/external-prometheus/`



6a. Understand how to install Linkerd with Helm

- ➔ Only gotcha here: you'll need to fetch `values-ha.yaml` for HA mode!
  - `helm fetch --untar linkerd/linkerd-control-plane`
  - Copy `linkerd-control-plane/values-ha.yaml` into your version control!
  - Once in version control, take a look.
    - It shouldn't need editing, but still, take a look.
  - `helm install … -f path/to/your/values-ha.yaml`

Installing Linkerd with Helm permalink:
`https://linkerd.io/2/tasks/install-helm/`



6b. Understand how to upgrade Linkerd with Helm

- ➔ **Always test** in non-production environments
- ➔ First upgrade the control plane, then (gradually) upgrade the data plane
  - ◆ Data plane is designed to work with future control planes
- ➔ **Don’t skip major versions**
  - ◆ ✓ 2.10.2 => 2.11.1 => 2.12.3 => 2.13.1
  - ◆ ❌ 2.10.2 => 2.13.1

Linkerd upgrade notes permalink:
`https://linkerd.io/2/tasks/upgrade/`



6b. Understand how to upgrade Linkerd with Helm

- Understand the `reuse-values` and `reset-values` Helm flags
- When no flag is used, the default behavior is:
  - `reuse-values` if there are overrides
  - `reset-values` if there are no overrides

	reuse-values	reset-values
With overrides	All values except overridden values will be reused	Nothing is reused and all values from the release are applied along with the overrides
Without overrides	All values will be reused	All values from the release are used

Upgrading Linkerd 2.13 with Helm:

`https://linkerd.io/2.13/tasks/install-helm/#upgrading-with-helm`



6c. Understand how to downgrade Linkerd with Helm

- This is why you want version control for your Helm stuff! 😀
- Fundamentally the same as upgrading Linkerd with helm
  - Any breaking changes will be documented in the helm upgrade guide
- **Gotcha**: there is no `helm downgrade` command
  - Use `helm upgrade --version` with an older version
  - Weird, but this is how Helm works 🤷

Upgrading Linkerd 2.13 with Helm:

`https://linkerd.io/2.13/tasks/install-helm/#upgrading-with-helm`



7. Understand how to debug Linkerd

At its core, this is “just” debugging a Kubernetes workload.

- ➔ `kubectl events` for any pod state other than running
- ➔ `kubectl logs` to view the container logs
- ➔ `kubectl get` will show you status information, too
  - ◆ This is especially relevant for Gateway API resources

Using `kubectl get` (since I needed a link for this slide 😂)

`https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#get`



7. Understand how to debug Linkerd

Don’t forget the Linkerd CLI commands:

- `linkerd check`
- `linkerd diagnostics proxy-metrics po/<pod-name> -n linkerd`
- `linkerd viz tap`
- `linkerd identity`
- `linkerd authz` (new in 2.11!)
- `linkerd diagnostics policy` (new in 2.13!)

Using `linkerd diagnostics`:
`https://linkerd.io/2.13/reference/cli/diagnostics/`



7. Understand how to debug Linkerd

You can set the proxy's log level:

- Globally with the Helm chart
  - Look for `logLevel` in `values.yaml` (be careful to get the one for the proxy)
- Per workload or namespace via annotation
  - `config.linkerd.io/proxy-log-level: warn,linkerd2_proxy=trace`
- At runtime!
  - `kubectl port-forward <pod-name> linkerd-admin &`
  - `curl -v --data 'linkerd=debug' -X PUT localhost:4191/proxy-log-level`

Changing the proxy log level:

`https://linkerd.io/2.13/tasks/modifying-proxy-log-level/`



7. Understand how to debug Linkerd

You can also set log levels for individual controllers:

- ➔ Globally with the Helm chart (`logLevel` values again)
- ➔ Per workload via Helm templates (link below)
  - ◆ or `kubectl edit` the Deployment directly
- ➔ Note that the Policy controller uses Rust levels
  - ◆ Helm `policyController.logLevel: linkerd=info,warn`
  - ◆ Others use Golang formats, e.g. `controllerLogLevel: error`

Helm templates:

`https://github.com/linkerd/linkerd2/blob/main/charts/linkerd-control-plane/templates/`



7. Understand how to debug Linkerd

Finally, there's the Linkerd debug sidecar:

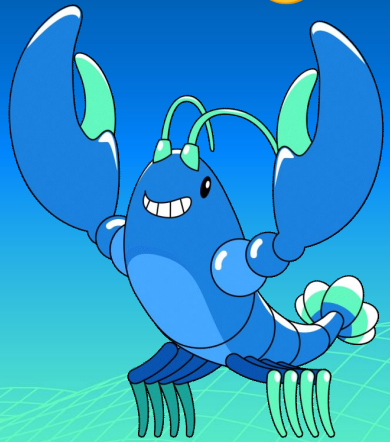
- Optional container with `tshark`, `tcpdump`, `lsof`, and `iproute2`
- Enabled by annotation, but must be manually injected into Linkerd control plane components
  - ```
    kubectl get deploy -n linkerd -o yaml \
        | linkerd inject --manual --enable-debug-sidecar \
        | kubectl apply -f -
    ```

Using the debug sidecar:
`https://linkerd.io/2/tasks/using-the-debug-container`

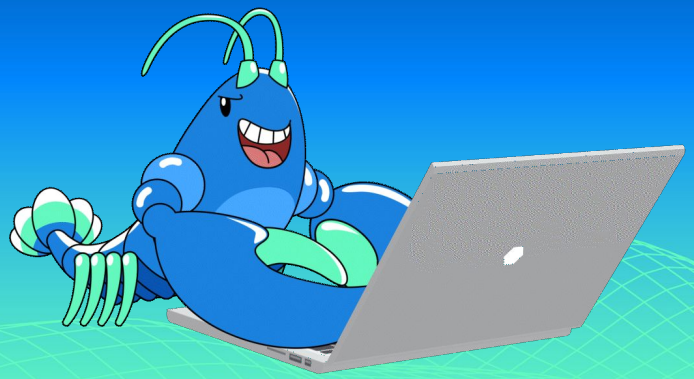

    ```

So that’s how it’s done.

…also, Buoyant Cloud can make much of this easier! 😀



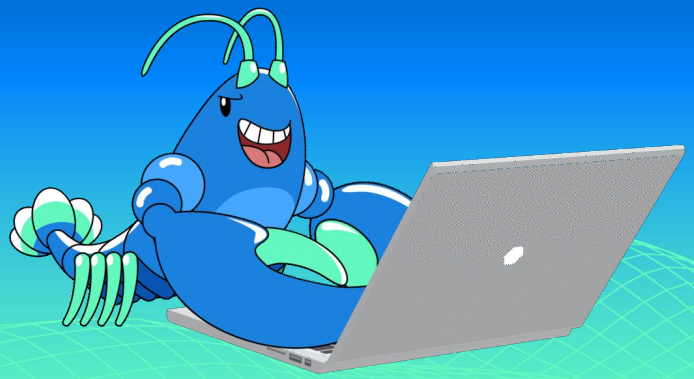
Workshop Time (almost)!



Workshop Caveats

- Several things we talked about above are simply out of scope for a workshop. 😐
  - Setting up a proper off-cluster CA
  - Using an external Prometheus
  - Putting configuration in version control
- We're mostly going to *show* the HA install with cert-manager managing identity issuer certs, plus a little Buoyant Cloud.
- We'll *tell* about other things.

Workshop Time!



🤞

Fully automated LINKERD on any Kubernetes cluster

Buoyant Cloud automates upgrades, data plane version tracking, mesh health alerts, and much, much more.

BOOK A DEMO
buoyant.io/demo



BUOYANT'S SERVICE MESH ACADEMY

The creators of LINKERD

Monthly hands-on, engineer-focused training from the creators of the service mesh

July 20: Linkerd Certificate Management Deep Dive

SIGN UP TODAY!
buoyant.io/sma



NEW

Join Buoyant's LINKERD Forum

Unlike Slack, messages will stick around and be searchable, increasing usefulness for everyone.

THE FORUM

linkerd.buoyant.io



BUOYANT'S SERVICE MESH ACADEMY

The creators of LINKERD

NEW!

Fundamentals of the Service Mesh Online Course + Certification

With hands-on LINKERD labs



BUOYANT
Creators of LINKERD

Thank you!

Flynn
Technical Evangelist, Buoyant
flynn@buoyant.io
@flynn on Slack

