BUOYANT

The creators of LINKERD

Real World GitOps with Argo and Linkerd



@BuoyantIO

buoyant.io

Regina Scott
Software Engineer, Red Hat



@reginacodes
@reginapizza
@Regina Scott

Flynn
Tech Evangelist, Buoyant



@kflynn
@Flynn

BUOYANT

LINKERD

What's on the agenda?





A bit of background first...

- ➔ Recap on our previous workshop (Argo, Linkerd, and Gateway API)

Then our workshop topics!

- ➔ Setting up Secrets Management
- ➔ Cert-manager
- ➔ Dealing with Linkerd Upgrades
- ➔ Dealing with Argo CD Upgrades
- ➔ Using Argo CD Sync Waves

LINKERD

Workshop Info

Follow along! **https://github.com/BuoyantIO/real-world-argo-linkerd**
Tools we'll use:

- ➔ Kubernetes cluster and the **kubectl** command
- ➔ **linkerd** – **https://linkerd.io/2/getting-started/**
- ➔ **argocd** – **https://argo-cd.readthedocs.io/en/stable/getting_started/**
- ➔ **kubeseal** – **https://github.com/bitnami-labs/sealed-secrets**
- ➔ **kubectl-argo-rollouts** – **https://argo-rollouts.readthedocs.io/en/stable/installation/**
- ➔ **step** – **https://smallstep.com/docs/step-cli/installation**
- ➔ **helm** – **https://helm.sh/docs/intro/quickstart/**
- ➔ **yq** – **https://github.com/mikefarah/yq?tab=readme-ov-file#install**
- ➔ **bat** – **https://github.com/sharkdp/bat**

Also recommended: **demosh** from **https://github.com/BuoyantIO/demosh**



BUOYANT

Creators of LINKERD

Join our Linkerd Forum!

Searchable, long-lived help threads and technical content.
(Slack is great, but messages die after 90 days.)

linkerd.buoyant.io

Buoyant Enterprise for LINKERD

Rust-based network security and reliability for modern applications. Built on open source and designed for the enterprise.

- ➔ Zero-trust security and compliance across your entire network
- ➔ Global traffic management and control
- ➔ Full L7 application observability
- ➔ Built for the enterprise

Learn more & get a **free trial** at buoyant.io/**enterprise-linkerd**

BUOYANT

Creators of LINKERD

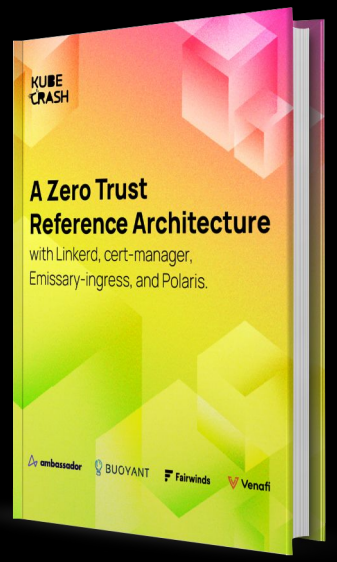
A Zero Trust Reference Architecture

with Linkerd, cert-manager, Emissary-ingress, and Polaris

60+ pages of highly technical open source content!

buoyant.io/**download/zero-trust-reference-architecture**

KUBE CRASH



BUOYANT

BUOYANT

SERVICE MESH ACADEMY

Creators of LINKERD

Get Certified! Fundamentals of the Service Mesh Online Course

With hands-on LINKERD labs at learn.buoyant.io

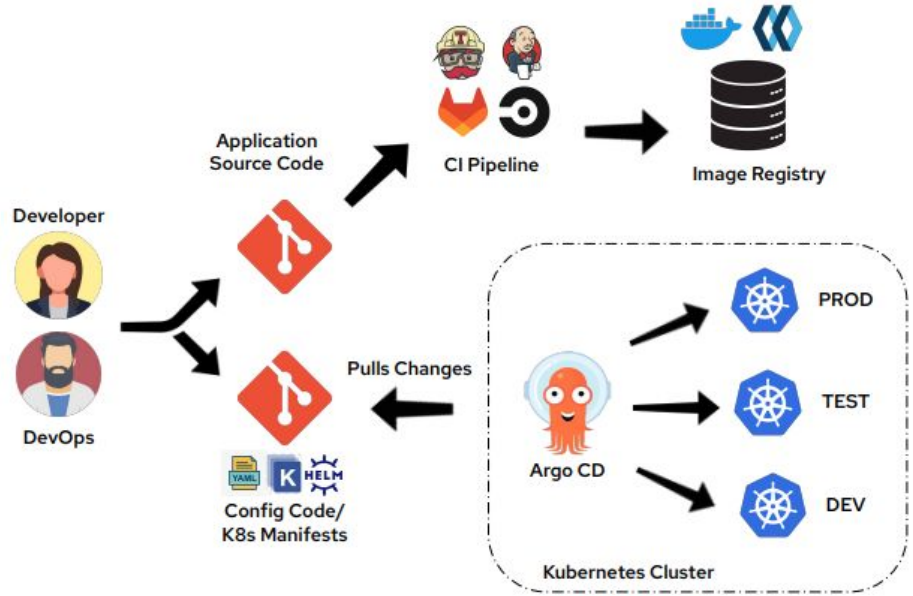


Argo CD



Argo CD Recap

- A Kubernetes deployment tool based on GitOps principles (uses Git repos as the source of truth)
- Argo CD is deployed *within* your K8s cluster
- Declarative K8s manifests are stored in git and can be yaml files, helm charts, kustomize, etc.
- When changes are detected in the git manifest repo, Argo CD pulls those changes and deploys the desired application states in the specified environments
- Argo CD can integrate with CI tools such as GitHub Actions and Jenkins



Benefits of Argo CD

- **Your entire Kubernetes configuration can be stored as code in git**
  - Config files are versioned and kept organized in git, as opposed to locally
  - Easy to rollback changes to a previous state if something goes wrong
  - Easy cluster disaster recovery
- Everyone who interacts with the configuration will use the same singular interface for interacting with and updating the cluster
- Argo CD will show the history of the changes
- Argo CD will compare the desired state to the actual state and can change it back to the desired state (this setting can be changed)
- No need to give cluster credentials anywhere outside of the K8's cluster
- Argo CD runs in one cluster, but can be set up to work with multiple clusters

Argo CD Sync Waves

- Sometimes in your cluster you want one thing deployed another
  - Common example: Database → Backend → Frontend
  - Our example: cert-manager → emissary → linkerd → faces app
- A “sync wave” is a mechanism to control the order of resource synchronization (aka deployment) during an application sync operation.
- All manifests have a sync wave of zero by default, but you can set these by using the `argocd.argoproj.io/sync-wave` annotation to add order to them
- The number you assign can be either positive or negative

Argo CD Sync Waves

- Argo CD will sync the lowest sync wave item first
- Argo CD will then make sure it returns a "Healthy" status before moving on, and will continue that way until all resources report healthy.
- Note that Argo CD will NOT apply the next manifest/perform the next sync until the previous sync waves all report healthy.
- You can have multiple resources in the same sync wave. In this case, Argo CD will choose what to sync first by Kind (Deployments, Services, Namespaces, etc.) and then by Name in ascending order.

The Problem with Sync Waves

- Sync waves only work for resources within the same application.
- This means that if you want separate applications, their sync waves won't know about each other and will not respect one another.
- One way around this is to use the App of Apps Pattern…



App-of-Apps Pattern

- Since an Argo CD Application is really just another Kubernetes resource, you can use an Application to deploy other Applications; this is what is known as the "App-of-Apps" pattern.
- In App-of-Apps, you have one parent Application that points to a folder with all your other Applications in it.
- Now when you label those child applications with sync-wave annotations, they will respect the sync wave order!

Important Note: App-of-Apps is an ADMIN-ONLY TOOL. You should really not have developers of your individual Applications able to modify the App-of-Apps.

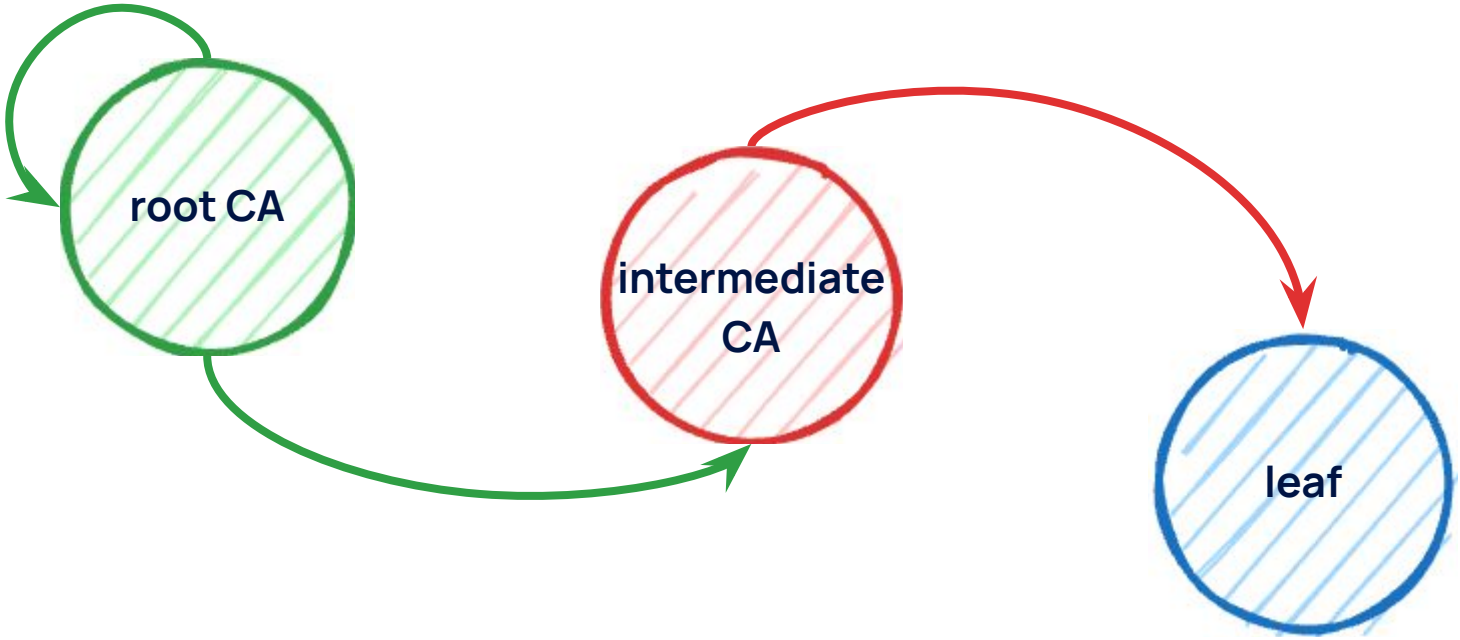
Linkerd, GitOps, and the Real World



Linkerd, GitOps, and the Real World

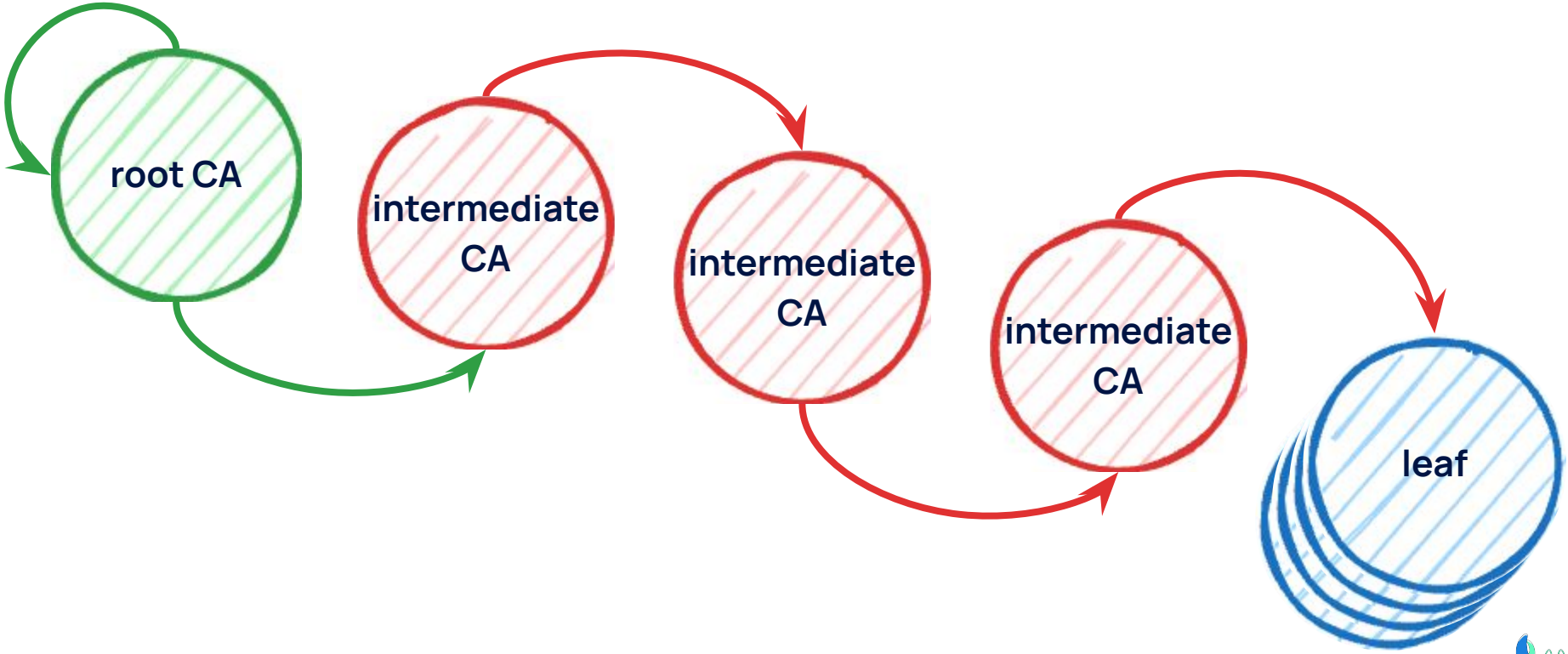
- When we introduced Argo CD and Linkerd in October, we didn't really show off GitOps.
- **This demo is all GitOps.**
  - Argo CD is managing just about everything, including Linkerd
  - Argo CD is using GitOps for everything it's managing
- **This demo is also more real-world.**
  - We're also going to use an external Vault and cert-manager to manage secrets for Linkerd.
  - We're going to use Bitnami Sealed Secrets to manage cert-manager's access to Vault.

Certificate Hierarchy Recap



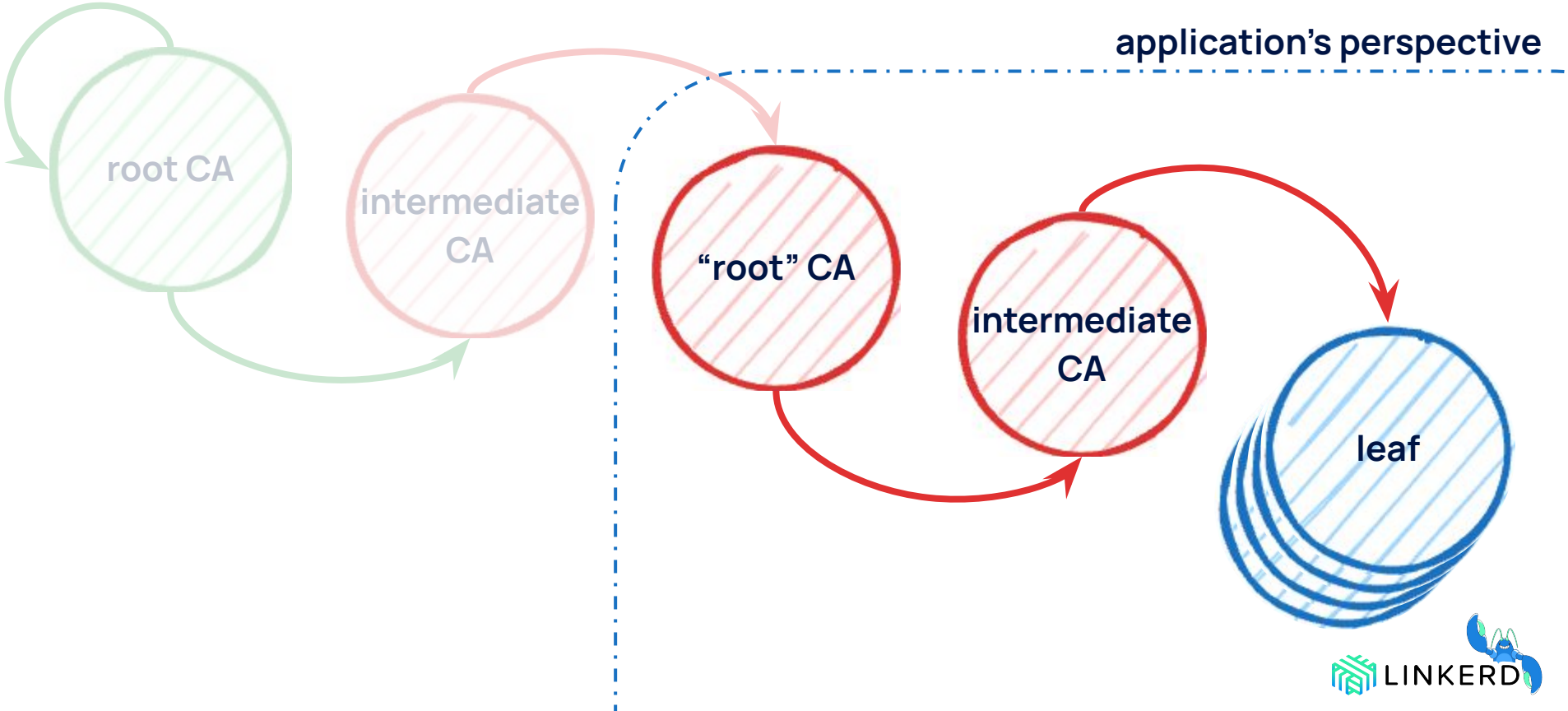
LINKERD

Certificate Hierarchy Recap

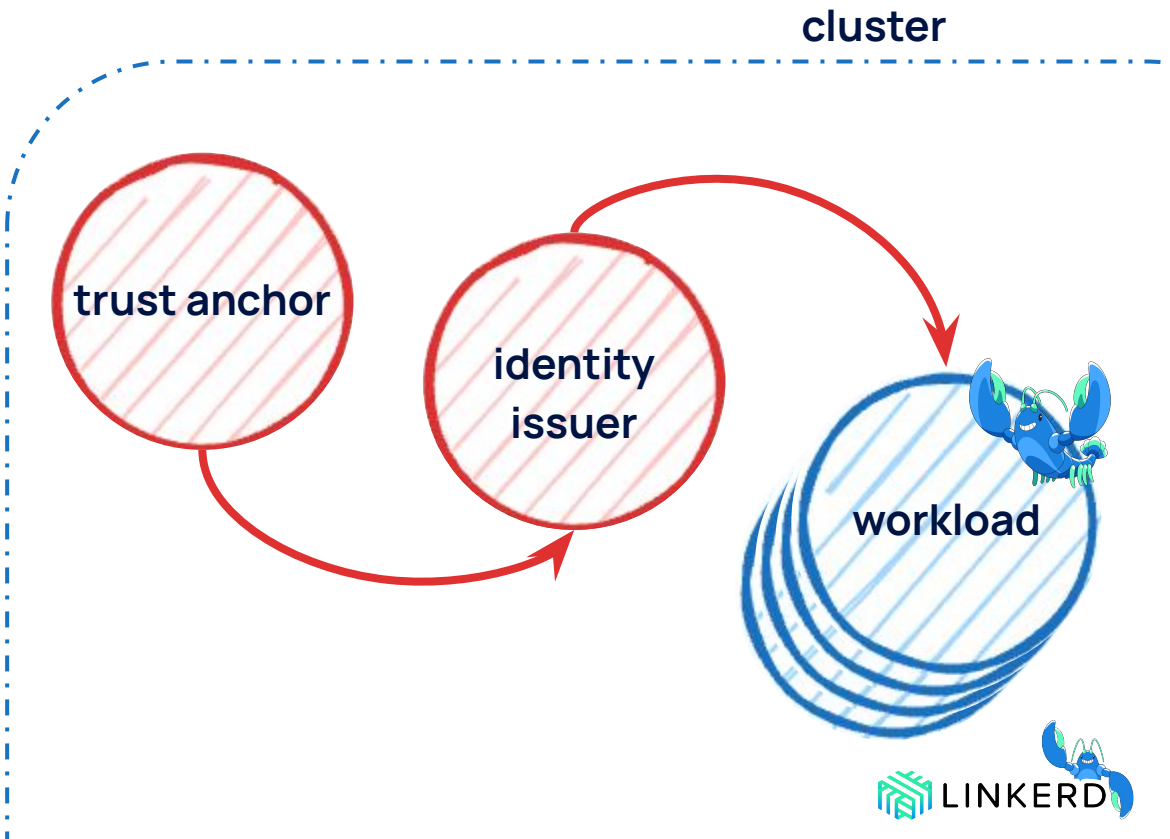


LINKERD

Certificate Hierarchy Recap: Delegation

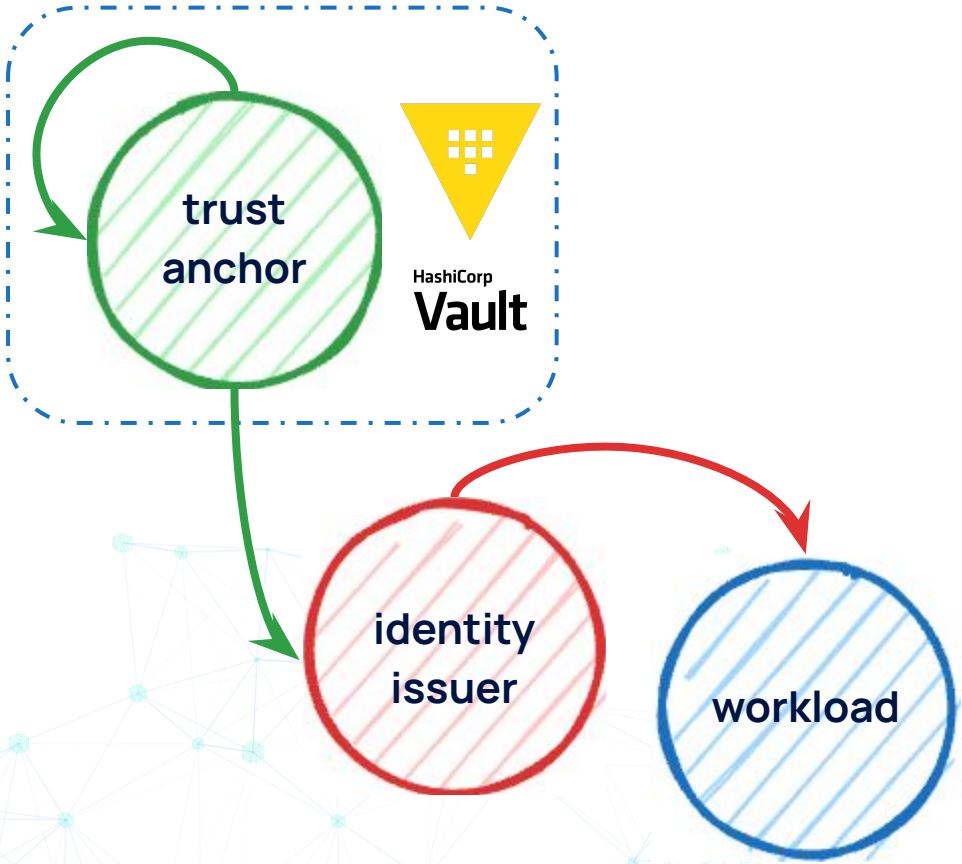


Certificate Hierarchy in Linkerd



Vault

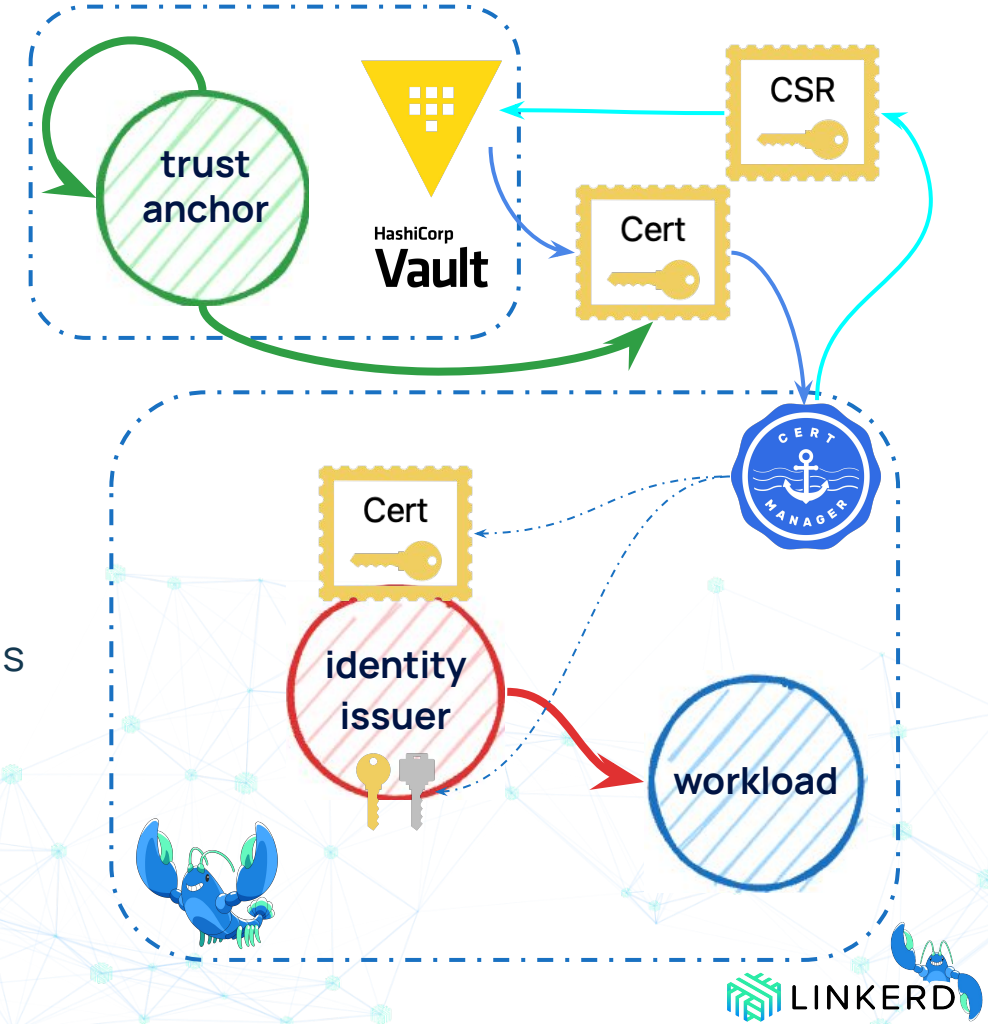
- Vault by HashiCorp, **`https://vaultproject.io/`**
  - "Manage secrets and protect sensitive data with Vault"
  - **NOT** a CNCF project 😢
- A basic function of Vault is being a safe place for private keys
- Vault can also act as a **CA**: it can use private keys it's storing to issue certificates, safely



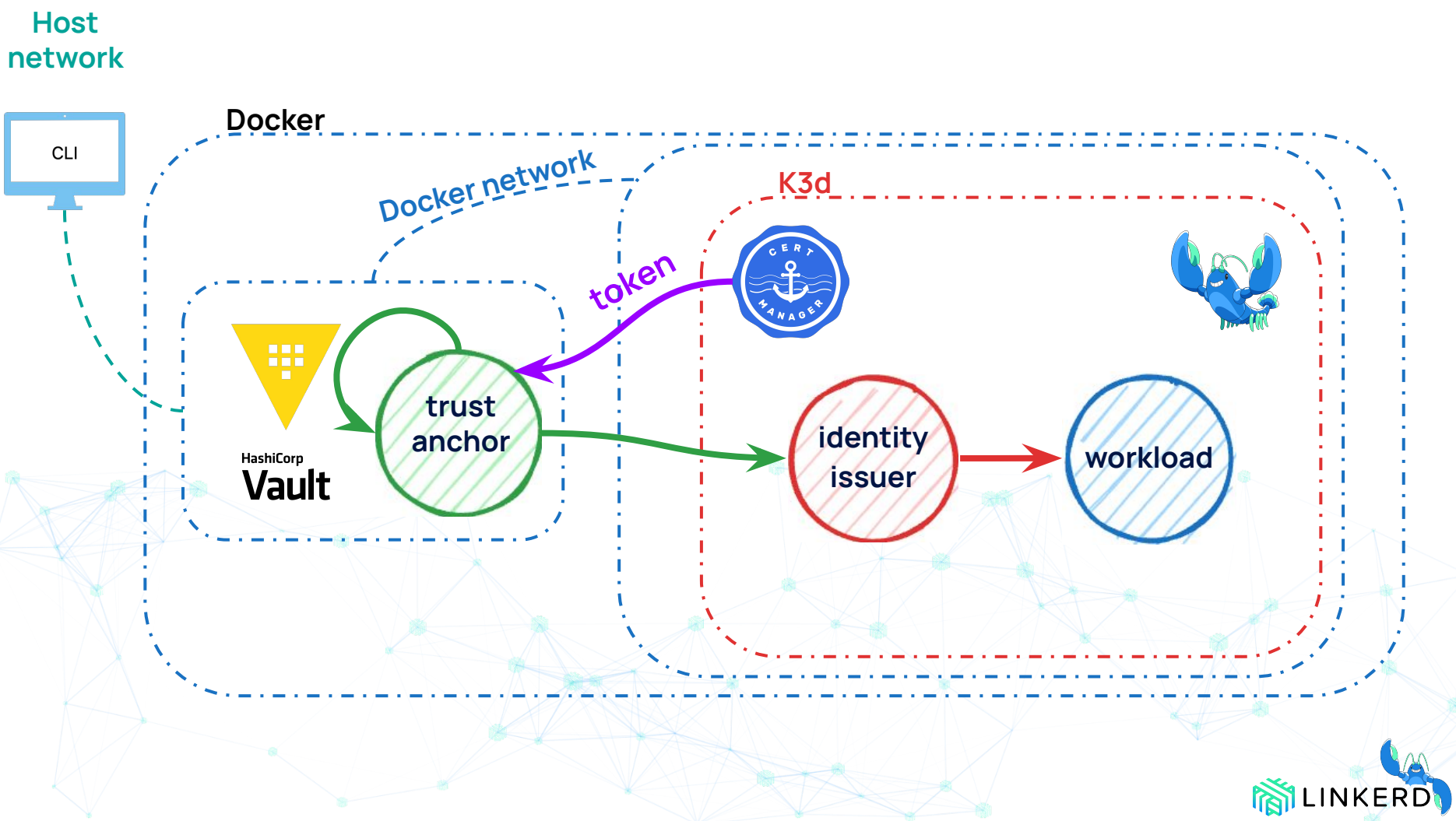
LINKERD

cert-manager

- cert-manager by Venafi, **`https://cert-manager.io/`**
  - "X.509 certificate management for Kubernetes and OpenShift"
  - A CNCF project 🙂
- Distributing and rotating certificates is a pain
- cert-manager automates much of this away
- Most powerful when paired with external CAs (like Vault)



LINKERD


Host network
CLI
Docker
Docker network
K3d
token
CERT MANAGER
HashiCorp Vault
trust anchor
identity issuer
workload
LINKERD

Secrets and GitOps

- **NEVER COMMIT SECRETS TO A GIT REPO.**
- The token that cert-manager uses to authenticate to Vault is sensitive, so it mustn't be committed as-is to Git.
- We have a **lot** of better options, as described at `https://argo-cd.readthedocs.io/en/stable/operator-manual/secret-management/`

Secrets and GitOps

- In this demo, we'll use Bitnami Sealed Secrets
  - https://github.com/bitnami-labs/sealed-secrets
- They rely on public-key cryptography, a controller, and a CLI:
  - Install the controller first
  - Use the CLI to encrypt your secret.
  - Save the *encrypted* secret in Git.
  - Let Argo do its thing with your Git manifests.
  - The controller "unseals" the Secret so your code can use it.
- We'll use this to permit cert-manager to access our Vault.

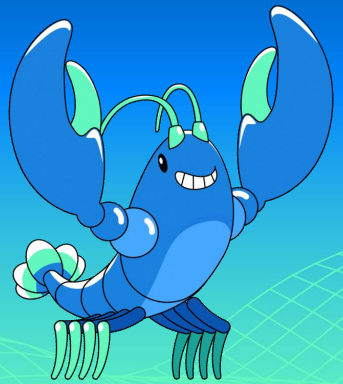
Secrets and Argo CD

Argo CD caches the manifests generated by plugins, along with the injected secrets, in its Redis instance. Those manifests are also available via the repo-server API (a gRPC service). This means that the secrets are available to anyone who has access to the Redis instance or to the repo-server.

Consider these steps to mitigate the risks of secret-injection plugins:

- ➔ Set up network policies to prevent direct access to Argo CD components (Redis and the repo-server). Make sure your cluster supports those network policies and can actually enforce them.
- ➔ Consider running Argo CD on its own cluster, with no other applications running on it.
- ➔ Enable password authentication on the Redis instance (currently only supported for non-HA Argo CD installations).

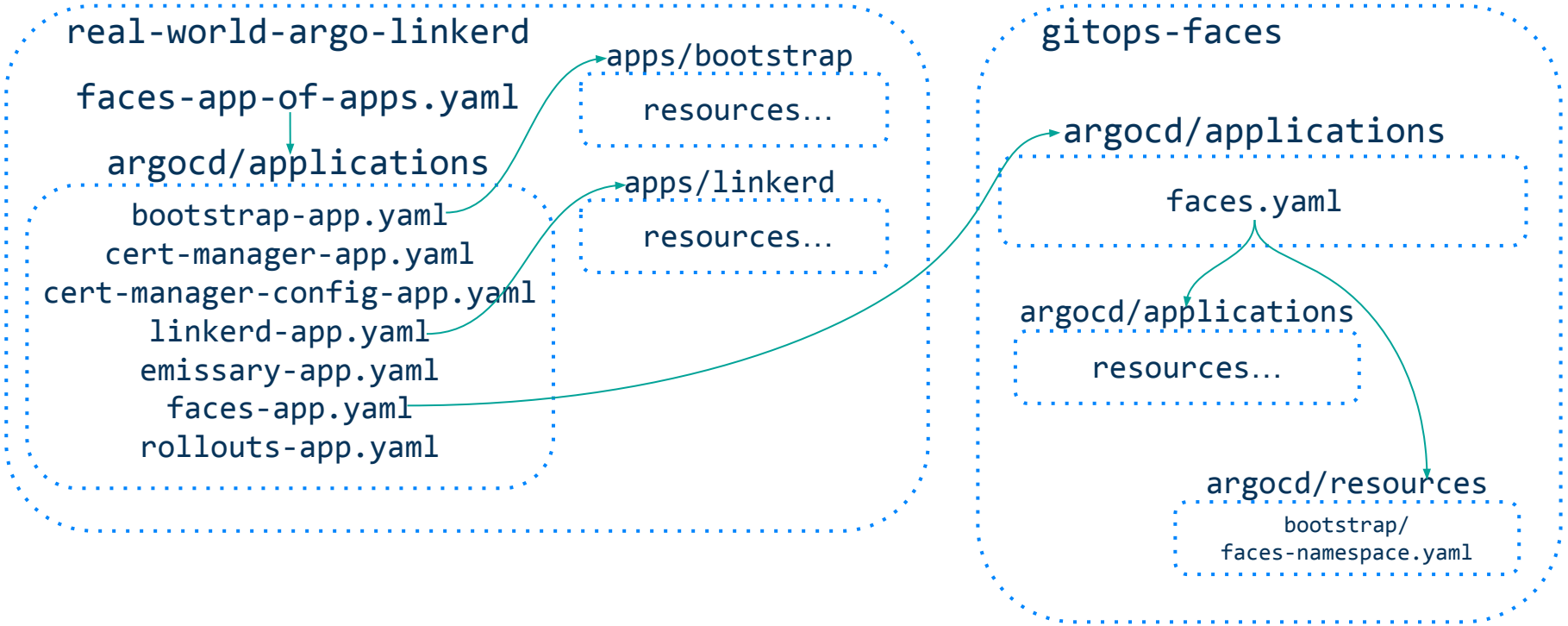
DEMO ARCHITECTURE



Demo Architecture

- We're using the Faces demo to show off GitOps
- Two repos:
  - **`https://github.com/BuoyantIO/real-world-argo-linkerd`** has the app-of-apps and infrastructure
  - **`https://github.com/BuoyantIO/gitops-faces`** has Faces itself and its configuration
- Why? It separates Faces from all the stuff you need to run and configure Faces.

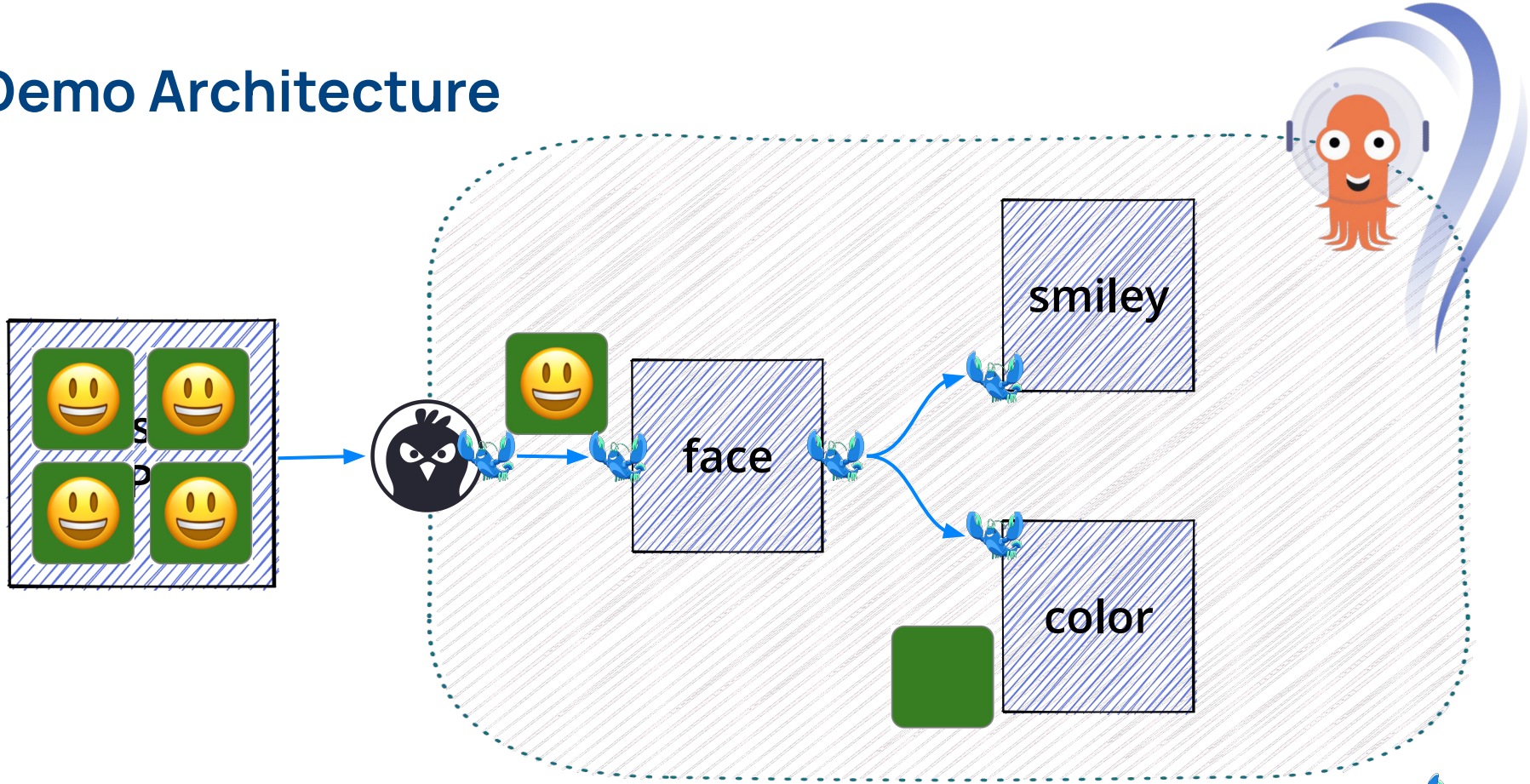
Demo Architecture



BUOYANT

LINKERD

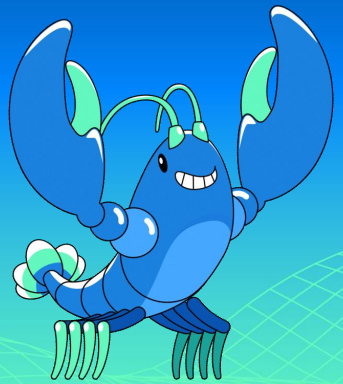
Demo Architecture



DEMO



Q & A





BUOYANT

SERVICE MESH ACADEMY

Creators of LINKERD

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

January 18: Hands-on with Linkerd Enterprise

SIGN UP TODAY!
buoyant.io/sma



BUOYANT'S
SERVICE MESH
ACADEMY

The creators of
LINKERD

Thanks so much!

Regina Scott, Argo CD Maintainer
rescott@redhat.com

Flynn, Tech Evangelist, Buoyant
flynn@buoyant.io

