LYRID

Unlocking the Power of Kubernetes

Handoyo Sutanto
CEO, Lyrid Inc.

About Handoyo

- Director of Solutions building multi-million dollar data centers and enterprise infrastructure software.
- Over 15 years of data center experience
- Past 4-5 years building:

Managing DevOps/ Engineering teams

Scalable enterprise software

DevOps infrastructures

What is Kubernetes?

In more technical terms, Kubernetes is an open-source orchestrator layer designed for automating the deployment, scaling, and operations of application containers across clusters of hosts.

To simplify this, Kubernetes is a tool that manages lots of small computer programs called containers. Containers are like tiny, self-contained packages that have everything a small part of a program needs to run. Kubernetes makes sure all these containers are running properly, can talk to each other, and can be updated or fixed easily if something goes wrong. It’s like a conductor for an orchestra, making sure every musician knows when to play their part in a big symphony.

Key Advantages

High Availability & Scalability

Kubernetes ensures that applications can scale and remain available to users even when servers face disruptions.

Portability & Flexibility

It can run on various infrastructures, including on-premises, hybrid, and public cloud environments.

Self-healing

It can automatically replace, reschedule, and restart containers that fail or do not meet user-defined health checks.

Automated Rollouts & Rollbacks

Kubernetes supports automated updates and rollbacks for applications, which helps maintain stability and reduce downtime.

In Simple Terms:

- Teamwork Master: Kubernetes is great at making sure all the small parts of a program work together nicely.
- Always Ready: It keeps programs ready to go, even if some parts need a quick fix or update.
- Grows with You: As your program gets more popular, Kubernetes can help it grow without skipping a beat.
- Smart Helper: It can fix problems on its own, like if a part of your program stops working, Kubernetes can restart it without you having to do anything.

The Benefits of Kubernetes

Make Your Application Robust

Self Healing Capabilities

The Container Advantage

Enhanced Security

Simplified Scalability

Kubernetes is Great… But…

Kubernetes Has a High Learning Curve

Companies can deploy Kubernetes without understanding every detail of how it works

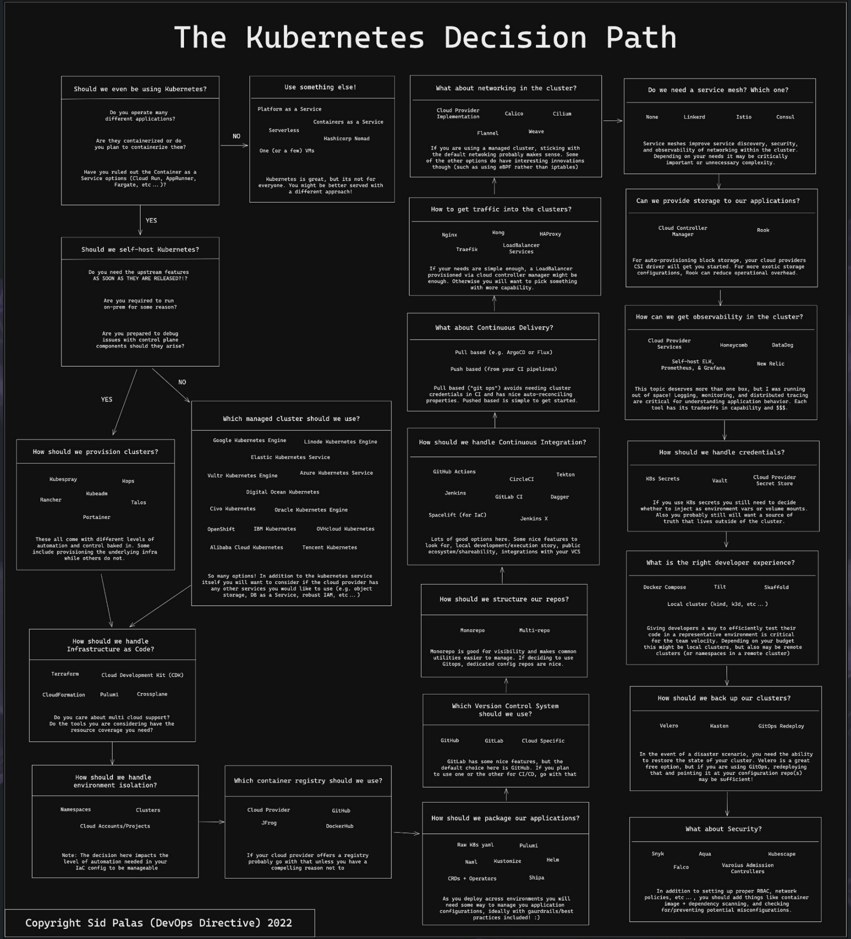
»

Lyrid Kubernetes enables developers to create Kubernetes native services.

Decision Fatigue

- The openness and flexibility of Kubernetes is a double edged sword.
- This image represents how much “choices” you will need to make to just “run” on Kubernetes.
- Image is from https://devopsdirective.com/

Interactive Path



The Kubernetes Decision Path

Should we even be using Kubernetes?

Do you operate many different applications?

Are they containerized or do you plan to containerize them?

Have you ruled out the Container as a Service options (Cloud Run, AppRunner, Fargate, etc ...)?

Use something else!

Platform as a Service

Containers as a Service

Serverless

Hashicorp Nomad

One (or a few) VMs

Kubernetes is great, but its not for everyone. You might be better served with a different approach!

Should we self-host Kubernetes?

Do you need the upstream features AS SOON AS THEY ARE RELEASED?!?

Are you required to run on-prem for some reason?

Are you prepared to debug issues with control plane components should they arise?

How should we provision clusters?

Kubespray

Kops

Rancher

Kubeadm

Talos

Portainer

These all come with different levels of automation and control baked in. Some include provisioning the underlying infra while others do not.

Which managed cluster should we use?

Google Kubernetes Engine

Linode Kubernetes Engine

Elastic Kubernetes Service

Vultr Kubernetes Engine

Azure Kubernetes Service

Digital Ocean Kubernetes

Civo Kubernetes

Oracle Kubernetes Engine

OpenShift

IBM Kubernetes

OVHcloud Kubernetes

Alibaba Cloud Kubernetes

Tencent Kubernetes

So many options! In addition to the kubernetes service itself you will want to consider if the cloud provider has any other services you would like to use (e.g. object storage, DB as a Service, robust IAM, etc ...)

How should we handle Infrastructure as Code?

Terraform

Cloud Development Kit (CDK)

CloudFormation

Pulumi

Crossplane

Do you care about multi cloud support? Do the tools you are considering have the resource coverage you need?

How should we handle environment isolation?

Namespaces

Clusters

Cloud Accounts/Projects

Note: The decision here impacts the level of automation needed in your IaC config to be manageable

Which container registry should we use?

Cloud Provider

GitHub

JFrog

DockerHub

If your cloud provider offers a registry probably go with that unless you have a compelling reason not to

How should we package our applications?

Raw K8s yaml

Pulumi

Naml

Kustomize

Helm

CRDs + Operators

Shipa

As you deploy across environments you will need some way to manage you application configurations, ideally with gaurdrails/best practices included! :)

Which Version Control System should we use?

GitHub

GitLab

Cloud Specific

GitLab has some nice features, but the default choice here is GitHub. If you plan to use one or the other for CI/CD, go with that

How should we structure our repos?

Monorepo

Multi-repo

Monorepo is good for visibility and makes common utilities easier to manage. If deciding to use Gitops, dedicated config repos are nice.

How should we handle Continuous Integration?

GitHub Actions

CircleCI

Tekton

Jenkins

GitLab CI

Dagger

Spacelift (for IaC)

Jenkins X

Lots of good options here. Some nice features to look for, local development/execution story, public ecosystem/shareability, integrations with your VCS

What about Continuous Delivery?

Pull based (e.g. ArgoCD or Flux)

Push based (from your CI pipelines)

Pull based ("git ops") avoids needing cluster credentials in CI and has nice auto-reconciling properties. Pushed based is simple to get started.

How to get traffic into the clusters?

Nginx

Kong

HAProxy

Traefik

LoadBalancer Services

If your needs are simple enough, a LoadBalancer provisioned via cloud controller manager might be enough. Otherwise you will want to pick something with more capability.

What about networking in the cluster?

Cloud Provider Implementation

Calico

Cilium

Flannel

Weave

If you are using a managed cluster, sticking with the default netwoking probably makes sense. Some of the other options do have interesting innovations though (such as using eBPF rather than iptables)

Do we need a service mesh? Which one?

None

Linkerd

Istio

Consul

Service meshes improve service discovery, security, and observability of networking within the cluster. Depending on your needs it may be critically important or unnecessary complexity.

Can we provide storage to our applications?

Cloud Controller Manager

Rook

For auto-provisioning block storage, your cloud providers CSI driver will get you started. For more exotic storage configurations, Rook can reduce operational overhead.

How can we get observability in the cluster?

Cloud Provider Services

Honeycomb

DataDog

Self-host ELK, Prometheus, & Grafana

New Relic

This topic deserves more than one box, but I was running out of space! Logging, monitoring, and distributed tracing are critical for understanding application behavior. Each tool has its tradeoffs in capability and $$$.

How should we handle credentials?

K8s Secrets

Vault

Cloud Provider Secret Store

If you use K8s secrets you still need to decide whether to inject as environment vars or volume mounts. Also you probably still will want a source of truth that lives outside of the cluster.

What is the right developer experience?

Docker Compose

Tilt

Skaffold

Local cluster (kind, k3d, etc ...)

Giving developers a way to efficiently test their code in a representative environment is critical for the team velocity. Depending on your budget this might be local clusters, but also may be remote clusters (or namespaces in a remote cluster)

How should we back up our clusters?

Velero

Kasten

GitOps Redeploy

In the event of a disaster scenario, you need the ability to restore the state of your cluster. Velero is a great free option, but if you are using GitOps, redeploying that and pointing it at your configuration repo(s) may be sufficient!

What about Security?

Snyk

Aqua

Kubescape

Falco

Varoius Admission Controllers

In addition to setting up proper RBAC, network policies, etc ... , you should add things like container image + dependency scanning, and checking for/preventing potential misconfigurations.

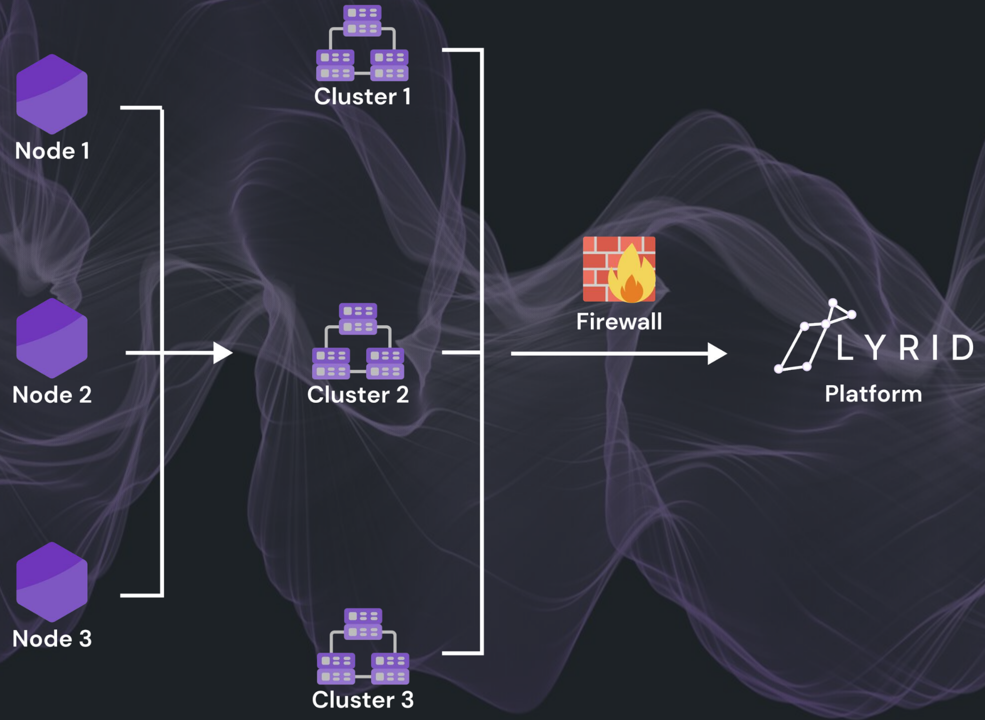
Copyright Sid Palas (DevOps Directive) 2022

You Need an Experienced In-House Team

Assembling an in-house team is a significant challenge

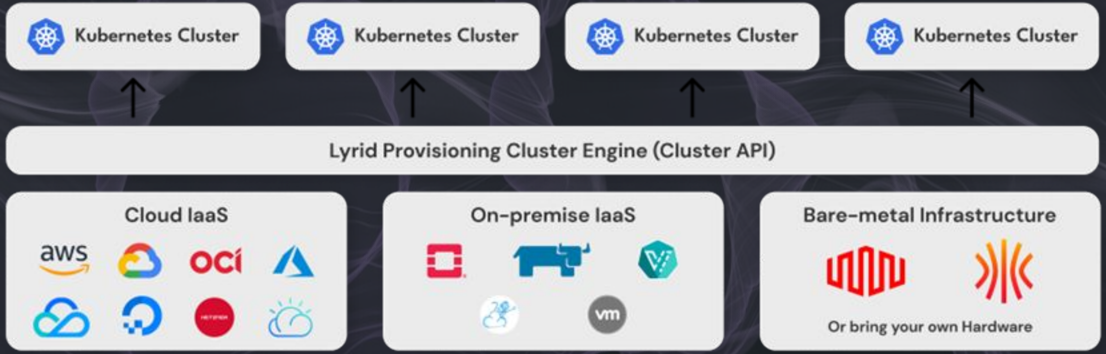
Lyrid's automated solutions handle the challenging parts of deployment and configuration

Lyrid Cloud Architecture



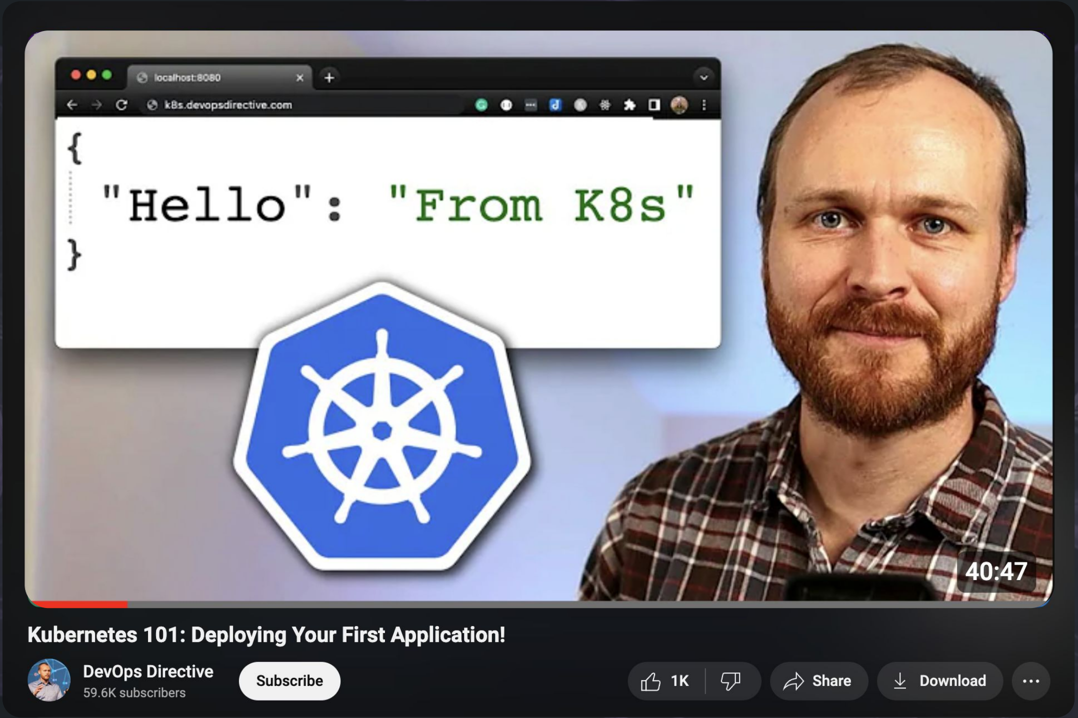
Demo - Lyrid Kubernetes Platform Partners

Powered by any infrastructures provider near you!



Example of Building and Serving Containers

- Your Code -> Container Packaging -> Registry -> Create Kubernetes Definitions (Deployment, Pod, Service) -> Create Load Balancer (Nginx) -> Create Ingress Definition -> Auto Certificate (LetsEncrypt) -> Rebuild and Redeploying (CI/CD)
- ~5 minutes of introduction, 30 minutes of instructions, and another 5 minutes of debugging
- And this is one of the fastest one that I encountered

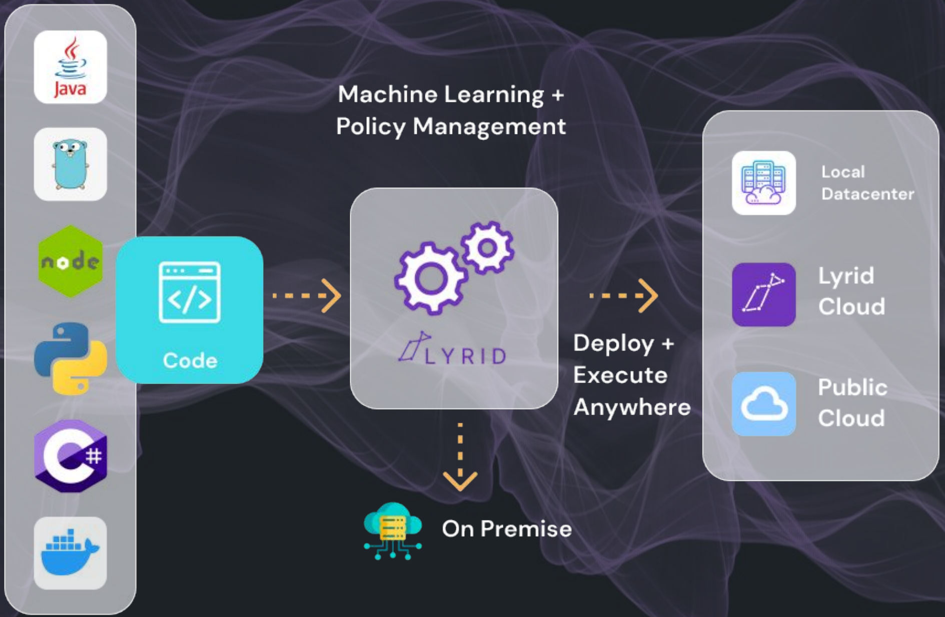


Kubernetes 101: Deploying Your First Application!

DevOps Directive
59.6K subscribers

https://www.youtube.com/watch?v=XltFOyGanYE

Lyrid Cloud Platform



01 Easy Deployments
You only need 3-clicks to deploy codes

02 One-Stop Manage Service Solution
We provide managed Kubernetes, databases and object storage

03 Lowers Barrier of Entry
You don't need to learn Kubernetes to run Kubernetes

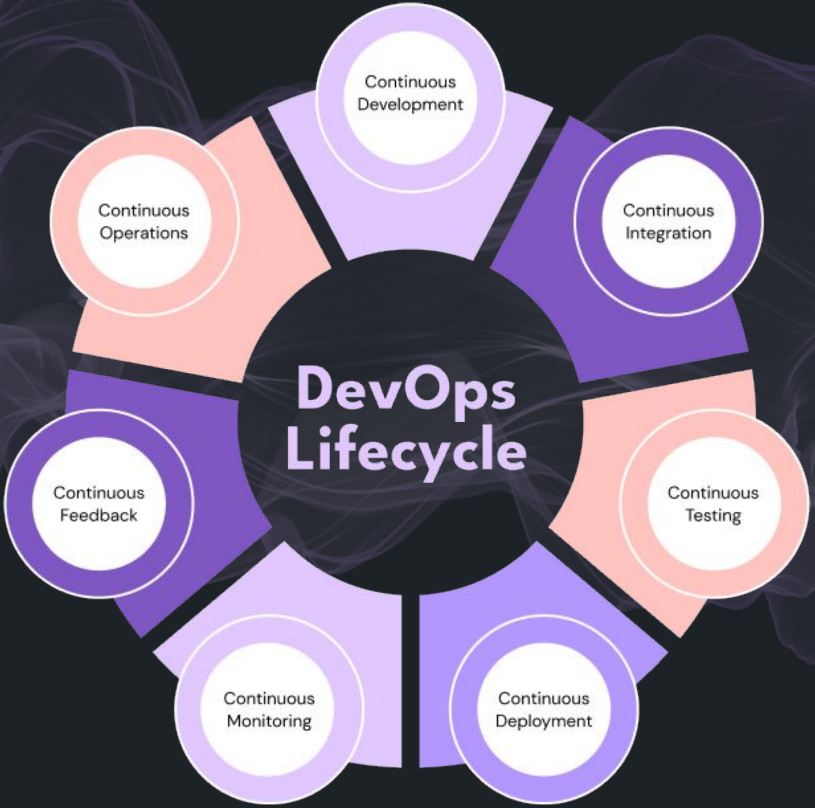
04 Developer Friendly
Eases your DevOps team to deploy codes and manage your microservices

05 Excellent Local Support
24/7 local support

Demo - Let’s see how we do fare

- One thing we do is that we prepackaged a lot of “best practices” that is shown
- Using the platform to show from code to publish
- And happens in the back

What is DevOps - Lifecycle Culture and Mindset



DevOps Lifecycle

Continuous Development

Continuous Integration

Continuous Testing

Continuous Deployment

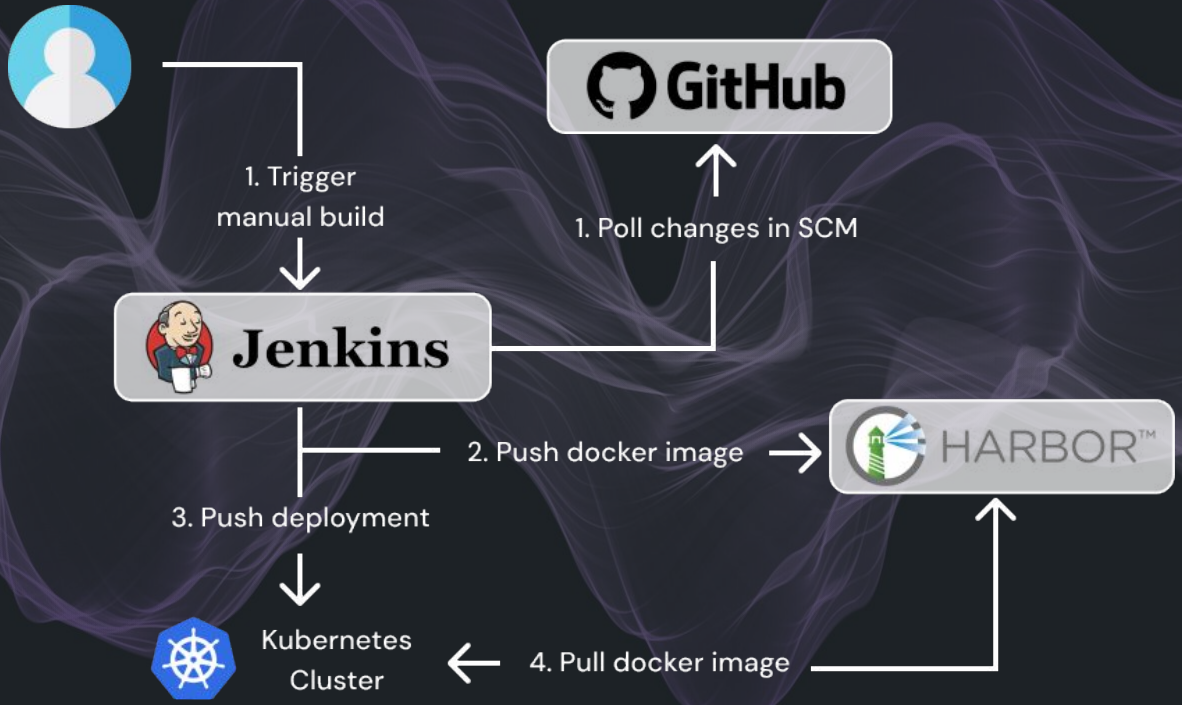
Continuous Monitoring

Continuous Feedback

Continuous Operations

LYRID

Just Go with the CI/CD Flow



GitHub

Jenkins

HARBOR™

Kubernetes Cluster

1. Trigger manual build

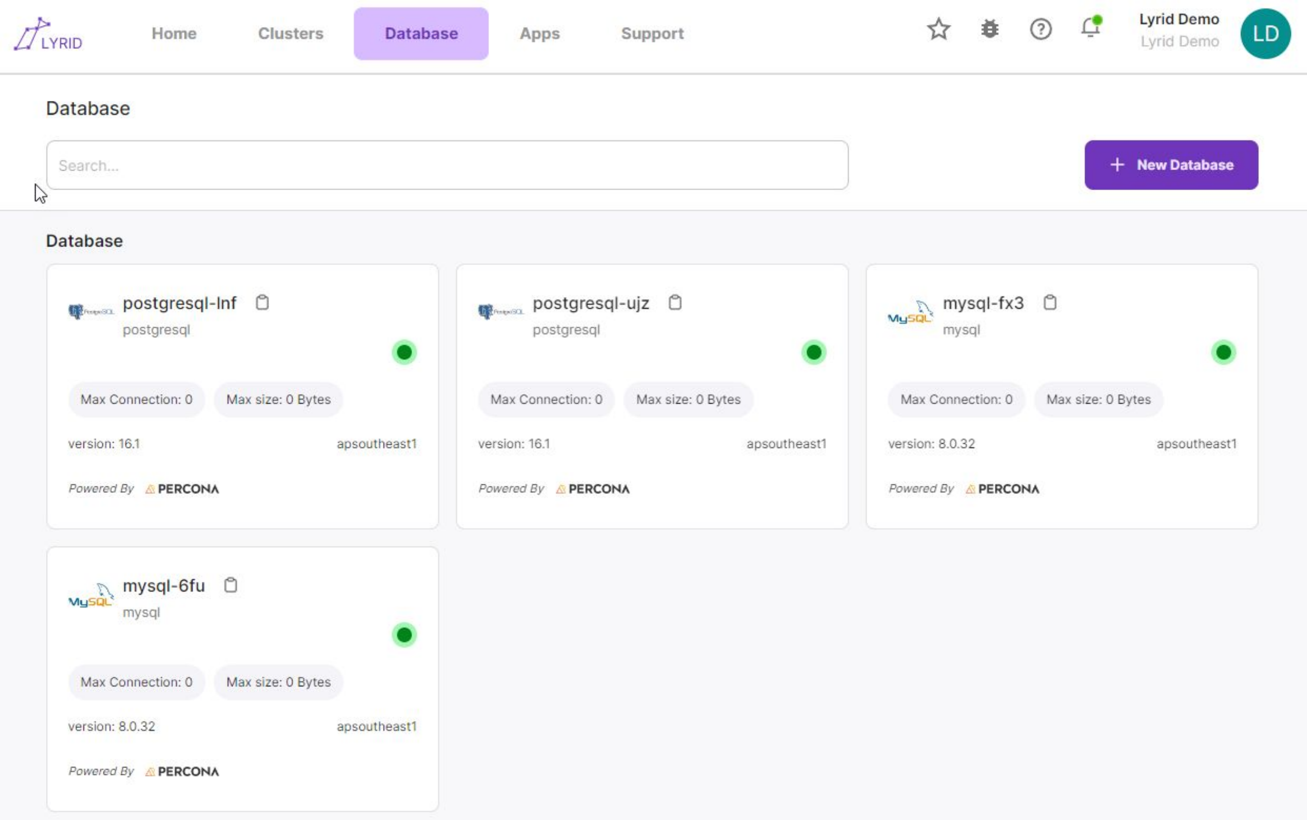
1. Poll changes in SCM

2. Push docker image

3. Push deployment

4. Pull docker image

Upcoming Features - Managed DB-as-a-Services



Database

Database

postgresql-Inf
postgresql
Max Connection: 0
Max size: 0 Bytes
version: 16.1
apsoutheast1
Powered By PERCONA

postgresql-ujz
postgresql
Max Connection: 0
Max size: 0 Bytes
version: 16.1
apsoutheast1
Powered By PERCONA

mysql-fx3
mysql
Max Connection: 0
Max size: 0 Bytes
version: 8.0.32
apsoutheast1
Powered By PERCONA

mysql-6fu
mysql
Max Connection: 0
Max size: 0 Bytes
version: 8.0.32
apsoutheast1
Powered By PERCONA

Contact

Handoyo Sutanto
CEO

hsutanto@lyrid.io
+1 408 660 6871

Hermon Christian Y. T.
VP of Business Development and Operations

hyobel@lyrid.io
+62 81 1305 8351

Address:

Jl. Pluit Indah 168B-G, Pluit Penjaringan,
Jakarta Utara, DKI Jakarta 14450

Socials:

hello@lyrid.io
www.lyrid.io
linkedin.com/company/lyrid
@lyridinc

Documentation:

https://docs.lyrid.io

LYRID

Sign up for Free!

Feedback Form

https://bit.ly/feedback-webinar-may2024

Step 1

Should we even be using Kubernetes?

Do you operate many different applications?

Are they containerized or do you plan to containerize them?

Have you ruled out the Container Service options (Cloud Run, AppRun, Fargate, etc…)?

YES

NO

PREVIOUS

Step 2 - NO

Use something else!

- Platform as a Service
- Containers as a Service
- Serverless
- Hashicorp Nomad
- One (or a few) VMs

NEXT

Kubernetes is great, but its not for everyone. You might be better served with a different approach!

PREVIOUS

Step 2 - YES

Should we self-host Kubernetes?

Do you need the upstream features AS SOON AS THEY ARE RELEASED?!?

Are you required to run on-prem for some reason?

Are you prepared to debug issues with control plane components should they arise?

YES

NO

PREVIOUS

Step 3 - YES

How should we provision clusters?

- Kubespray
- Kops
- Kubeadm
- Rancher
- Portainer
- Talos

NEXT

These all come with different levels of automation and control baked in. Some include provisioning the underlying infra while others do not.

PREVIOUS

Step 3 - NO

How should we provision clusters?

- Google Kubernetes Engine
- Linode Kubernetes Engine
- Elastic Kubernetes Service
- Vultr Kubernetes Engine
- Azure Kubernetes Service
- Digital Ocean Kubernetes

- Civo Kubernetes
- Oracle Kubernetes Engine
- OpenShift
- IBM Kubernetes
- OVHcloud Kubernetes
- Alibaba Cloud Kubernetes
- Tencent Kubernetes

NEXT

So many options! In addition to the kubernetes service itself you will want to consider if the cloud provider has any other services you would like to use (e.g object storage, DB as a service, robust IAM, etc…)

PREVIOUS

Step 4

How should we handle infrastructure as Code?

- Terraform
- Cloud Development Kit (CDK)
- CloudFormation
- Pulumi
- Crossplane

NEXT

Do you care about multi cloud support?

Do the tools you are considering have the resource coverage you need?

PREVIOUS

Step 5

How should we handle environment isolation?

- Namespaces
- Clusters
- Cloud Accounts/Projects

Note: The decision here impacts the level of automation needed in your IaC confid to be manageable

NEXT

PREVIOUS

Step 6

Which container registry should we use?

- Cloud Provider
- GitHub
- JFrog
- DockerHub

If your cloud provider offers a registry probably go with that unless you have a compelling reason not to

NEXT

PREVIOUS

Step 7

How should we package our application?

- Raw K8s yaml
- Pulumi
- Naml
- Kustomize
- Helm
- CRDs + Operators
- Shipa

NEXT

As you deploy across environments you will need some way to manage your application configurations, ideally with guardrails/best practices included! :)

PREVIOUS

Step 8

Which Version Control System should we use?

- GitHub
- GitLab
- Cloud Specific

NEXT

GitLab has some nice features, but the default choice here is GitHub. If you plan to use one or the other for CI/CD, go with that

PREVIOUS

Step 9

How should we structure our repos?

- Monorepo
- Multi-repo

NEXT

Monorepo is good for visibility and makes common utilities easier to manage. If deciding to use Gitops, dedicated config repos are nice

PREVIOUS

Step 10

How should we handle Continuous Integration?

- GitHub Actions
- Tekton
- CircleCI
- Jenkins
- GitLab CI
- Dagger
- Spacelift (for IaC)
- Jenkins X

NEXT

Lots of good options here. Some nice features to look for, local development/ execution story, public ecosystem/shareability, integrations with your VCS

PREVIOUS

Step 11

What about Continuous Delivery?

NEXT

- Pull based (e.g. ArgoCD or Flux)
- Push based (from your CI pipelines)

Pull based (“git ops”) avoids needing cluster credentials in CI and has nice auto-reconciling properties. Pushed based is simple to get started.

PREVIOUS

Step 12

How to get traffic into the clusters?

- Nginx
- Kong
- HAProxy
- Traefik
- LoadBalancer Services

NEXT

If your needs are simple enough, a LoadBalancer provisioned via cloud controller manager might be enough. Otherwise you will want to pick something with more capability.

PREVIOUS

Step 13

What about networking in the cluster?

- Cloud Provider Implementation
- Calico
- Cilium
- Flannel
- Weave

NEXT

If you are using a managed cluster, sticking with the default networking probably makes sense. Some of the other options do have interesting innovations though (such as using eBPF rather than iptables)

PREVIOUS

Do we need a service mesh? Which one?

- None
- Linkerd
- Istio
- Consul

NEXT

Service meshes improve service discovery security, and observability of networking within the cluster. Depending on your needs it may be critically important or unnecessary complexity.

PREVIOUS

Step 15

Can we provide storage to our applications?

NEXT

- Cloud Controller Manager
- Rook

For auto-provisioning block storage, your cloud providers CSI driver will get you started. For more exotic storage configurations, Rook can reduce operational overhead.

PREVIOUS

How can we get observability in the cluster?

- Cloud Provider Services
- Honeycomb
- DataDog
- Self-host ELK, Prometheus, & Grafana
- New Relic

NEXT

This topic deserves more than one box, but I was running out of space! Logging, monitoring, and distributed tracing are critical for understanding application behavior. Each tool has its tradeoffs in capability and $$$.

PREVIOUS

Step 17

How should we handle credentials?

- K8s Secrets
- Vault
- Cloud Provider Secret Store

NEXT

If your use K8s secrets you still need to decide whether to inject as environment vars or volume mounts. Also you probably still will want a source of truth that lives outside of the cluster.

PREVIOUS

Step 18

How should we backup our clusters?

- Velero
- Kasten
- GitOps Redeploy

NEXT

In the event of a disaster scenario, you need the ability to restore the state of your cluster. Velero is great free option, but if you are using GitOps, redeploying that and pointing it at your configuration repo(s) may be sufficient!

PREVIOUS

What about Security?

- Snyk
- Aqua
- Kubescape
- Falco
- Various Admission Controllers

In addition to setting up proper RBAC, network policies, etc…, you should add things like container image + dependency scanning, and checking for/preventing potential misconfigurations.

NEXT

PREVIOUS

You’re All Set!

Back to Slide 8

PREVIOUS