Production Ready Checklists for Kubernetes

weaveworks

Production Ready Checklists for Kubernetes

Production readiness is a term you hear a lot, and depending on who you are talking to and what they are doing, it can mean different things.

Readiness is dependant on your use case and can be about making tradeoffs. Although a cluster can be production ready when it's good enough to serve traffic, many agree that there are a minimum set of requirements you need before you can safely declare your cluster ready for commercial traffic.

The first checklist describes all of the things you need to include in your app before it gets deployed to Kubernetes. The second checklist outlines the infrastructure you need to have in place before running production traffic on a cluster.

APPLICATION CHECKLIST FOR KUBERNETES

These are the checks and balances needed before an app is deployed to Kubernetes.

	What is it?	Why you need it?	Options
Readiness check	Endpoints for Kubernetes to monitor your application lifecycle.	Allows Kubernetes to restart or stop traffic to a pod. Readiness failure is transient and tells Kubernetes to route traffic elsewhere. Readiness failure is useful for startup and load management.	Read the post: **Resilient apps with Liveness and Readiness probes**
Liveness check	Endpoints for Kubernetes to monitor your application lifecycle.	Liveness failure is for telling Kubernetes to restart the pod.	Read: **Resilient apps with Liveness and Readiness probes**
Metric instrumentation	Code and libraries used in your code to expose metrics.	Allows measuring operation of application and enables many more advanced use cases.	Prometheus, New Relic, Datadog and others. Read: **Monitoring Kubernetes with Prometheus**
Dashboards	View of metrics.	You need to understand the data.	Grafana and many other options.
Playbooks and Runbooks	Rich guides for your engineers on how-to operate the system and fault find when things go wrong.	Nobody is at their sharpest at 03:00 AM. Knowledge deteriorates over time.	Confluence. Markdown files. Weave Cloud Notebooks.
Limits and requests	Explicit resource allocation for pods.	Allows Kubernetes to make good scheduling decisions.	Read: **Kubernetes Pod Resource Limitations and Quality of Service**
Labels and annotations	Metadata held by Kubernetes.	Makes workload management easier and allows other tools to work with standard Kubernetes definitions.	Read: **Labels and Selectors in Kubernetes**

	What is it?	Why you need it?	Options
Alerts	Automated notifications on defined trigger.	You need to know when your service degrades.	Prometheus and Alertmanager. Read: **Labels in Prometheus alerts: think twice before using them**
Structured logging output	Output logs in a machine readable format to facilitate searching and indexing.	Trace what went wrong when something does.	**ELK stack** (Elasticsearch, Logstash and Kibana). Many commercial offerings.
Tracing instrumentation	Instrumentation to send request processing details to a collection service.	Sometimes the only way of figuring out where latency is coming from.	**Zipkin**, **Lightstep**, **Appdash**, **Tracer**, **Jaeger**
Graceful shutdowns	Applications respond to SIGTERM correctly.	This is how Kubernetes will tell your application to end.	Read: **10 tips for Building and Managing Containers**
Graceful dependency (w. Readiness check)	Applications don't assume dependencies are available. Wait for other services before reporting ready.	Avoid headaches that come with a service order requirement.	Read: **10 tips for Building and Managing Containers**
Configmaps	Define a configuration file for your application in Kubernetes using configmaps.	Easy to reconfigure an app without rebuilding, allows config to be versioned.	Read: **Best Practices for Designing and Building Containers for Kubernetes**
Labeled images using commit SHA	Label the docker images with the code commit SHA.	Makes tracing image to code trivial.	
Locked down runtime context	Use deliberately secure configuration for application runtime context.	Reduces attack surface, makes privileges explicit.	

THE CLUSTER READY CHECKLIST

These are the items that need to be in place for your cluster before running it in production.

	What is it?	Why you need it?	Options
Build pipeline	The CI portion of the CICD pipeline. Tests, integrates and builds your container artefact. Artefacts should be tagged with the Git commit SHA to verify provenance.	To deposit clean build artefact to Container Registry.	**CircleCI**, **Travis** and **Jenkins** and others.
Deployment pipeline	The deployment portion of the CICD. Takes the build artefacts, and delivers them to the cluster. This is where GitOps occurs.	More secure way of doing deployment. Can add approval process if necessary.	**Weave Cloud**, and Flux
Image registry	Stores build artefacts. Need credentials for CI to push and for cluster to pull images.	Keep versioned artefacts available.	Roll your own. Commercial: **DockerHub**, **JFrog**, or **GCP Registry**.
Monitoring infrastructure	Collects and stores metrics.	Understand your running system. Get alerts when something goes wrong.	OSS: **Prometheus**, Cortex, Thanos. Commercial: Datadog, **Grafana Cloud**, **Weave Cloud**
Shared storage	Store persistent state of your application beyond the pod's lifetime.	No one has a stateless app.	Many. Depends on the platform.
Secrets management	How do your applications access secret credentials, securely?	Secrets are required to access external services.	**Bitnami Sealed Secrets** **Hashicorp Vault**
Ingress controller	Common routing point for all inbound traffic.	Easier to manage authentication and logging	Platform controller (**AWS ELB**) **Google Compute Engine** (GCE) & **NGINX** (Kubernetes) **Kong**, **Traefik**, **HAProxy**, **Istio**, **Envoy**

	What is it?	Why you need it?	Options
API Gateway	Single point for incoming requests. Higher layer ingress controller that can replace an ingress controller.	Can route at HTTP level. Enables common and centralised tooling for tracing, logging, authentication.	**Ambassador (Envoy)**. Roll your own
Service mesh	Additional layer on top of Kubernetes to manage routing.	Enables complex use cases like Progressive Delivery. Adds interservice TLS, load balancing, service discovery, monitoring and tracing.	**Linkerd** **Istio**
Service catalogue / Broker	Enables easy dependencies on services and service discovery for your team.	Simplifies deploying applications.	Kubernetes' **service catalog API**
Network policies	Rules on allowed connections and services. Needs a CNI plugin.	Prevent unauthorised access, improve security, segregate namespaces.	**Weave Net** Calico
Authorization integration	API level integration into the Kubernetes auth flow.	Uses existing SSO to reduce the number of accounts and to centralize account management. Can integrate with almost any auth provider.	Requires custom work.
Image scanning	Automated scanning of vulnerability in your container images. Implement in the CI pipeline.	Because CVEs happen.	**Docker, Snyk, Twistlock, Sonatype, Aqua Security**
Log aggregation	Bring all logs from application into a searchable place.	Logs are the best source of information on what went wrong.	Many options. **Fluentd** or **ELK** (Elasticsearch, Logstash, Kibana) stack are good bets for roll-your-own

HAVE QUESTIONS ON WHAT YOU NEED TO CREATE A CLOUD NATIVE PLATFORM?

Our Enterprise Kubernetes Platform reduces complexity through configuration management, automation and operations tooling. With GitOps configuration management, teams can define a standard installation of Kubernetes and automate the deployment of new nodes with a predefined template. It also simplifies additional cluster installations, since Kubernetes is already preconfigured and any applications running on top of it are also predefined.

Find out more about the **Enterprise Kubernetes Platform** or **request a demo**.