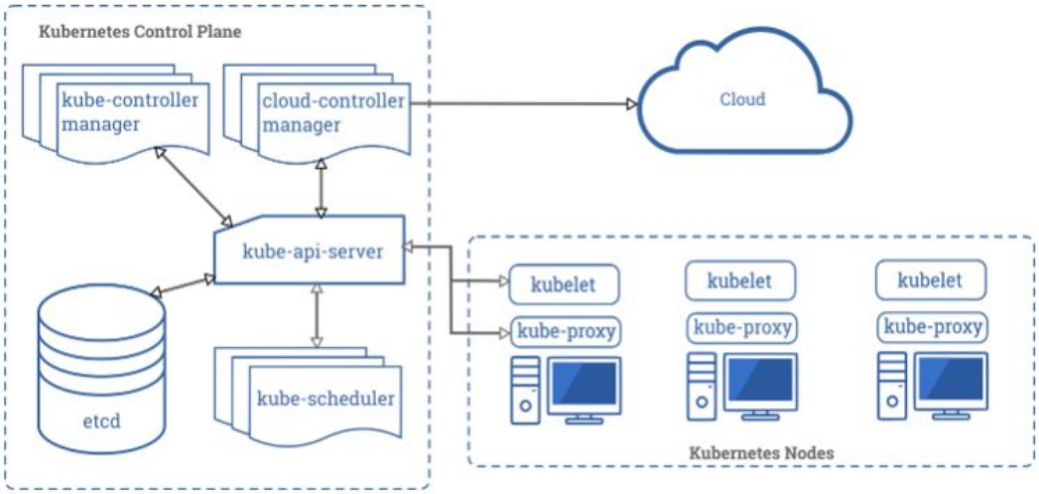
Probes

Pod Lifecycle



Pod Lifecycle

Status	Description
`Pending`	The Pod has been accepted by the Kubernetes cluster, but one or more of the containers has not been set up and made ready to run. This includes time a Pod spends waiting to be scheduled as well as the time spent downloading container images over the network.
`Running`	The Pod has been bound to a node, and all of the containers have been created.
`Succeeded`	All containers in the Pod have terminated in success, and will not be restarted.
`Failed`	All containers in the Pod have terminated, and at least one container has terminated in failure.
`Unknown`	For some reason the state of the Pod could not be obtained.

Pod Conditions

Condition	Description
`PodScheduled`	The Pod has been scheduled to a node.
`Initialized`	All init containers have completed successfully.
`ContainersReady`	All containers in the Pod are ready.
`Ready`	The Pod is able to serve requests and should be added to the load balancing pools of all matching Services.

Probes

A **probe** is a diagnostic performed periodically by the kubelet on a container.

The kubelet can optionally perform and react to **three** kinds of probes on running containers.

- **livenessProbe**
- **readinessProbe**
- **startupProbe**

livenessProbe

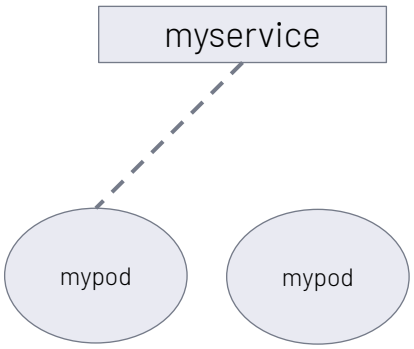
- Many applications running for long periods of time eventually transition to broken states, and cannot recover except by being restarted. Kubernetes provides liveness probes to detect and remedy such situations.
- If the process in your container is able to crash on its own whenever it encounters an issue or becomes unhealthy, you do not necessarily need a liveness probe; the kubelet will automatically perform the correct action in accordance with the Pod's restartPolicy.
- If the liveness probe fails, the kubelet kills the container, and the container is subjected to its restart policy.

readinessProbe

- Sometimes, applications are temporarily unable to serve traffic. For example, an application might need to load large data or configuration files during startup, or depend on external services after startup.
- In such cases, you don't want to kill the application, but you don't want to send it requests either. Kubernetes provides readiness probes to detect and mitigate these situations.

readinessProbe

- If you'd like to start sending traffic to a Pod only when a probe succeeds, specify a readiness probe.
- If the readiness probe fails, the endpoints controller removes the Pod's IP address from the endpoints of all Services that match the Pod.



startupProbe

- Sometimes, you have to deal with legacy applications that might require an additional startup time on their first initialization.
- Startup probes are useful for Pods that have containers that take a long time to come into service.
- All other probes are disabled if a startup probe is provided, until it succeeds.

Check mechanisms

There are four different ways to check a container using a probe.

- **exec**
- **httpGet**
- **tcpSocket**
- **grpc**

Check mechanisms

- **exec**

  Executes a specified command inside the container. The diagnostic is considered successful if the command exits with a **status code of 0**.

- **httpGet**

  Performs an **HTTP GET request** against the Pod's IP address on a specified port and path. The diagnostic is considered successful if the response has a status code **greater than or equal to 200 and less than 400.**

Check mechanisms

- **tcpSocket**

  With **tcpSocket**, the kubelet will attempt to open a socket to your container on the specified port. If it can establish a connection, the container is considered healthy, if it can't it is considered a failure.

- **grpc**

  If your application implements the **gRPC Health Checking Protocol**, You can use it for application liveness checks. It performs a **remote procedure call** using **gRPC**.

Probe outcome

Each probe has one of three results:

- **Success:** The container passed the diagnostic.
- **Failure:** The container failed the diagnostic.
- **Unknown:** The diagnostic failed (no action should be taken, and the kubelet will make further checks).

THANKS!

Any questions?

Container States

State	Description
`Waiting`	If a container is not in either the Running or Terminated state, it is Waiting.
`Running`	The Running status indicates that a container is executing without issues.
`Terminated`	A container in the "Terminated" state is a container that has either completed its execution or terminated due to an error.