# Choosing the Right Kubernetes Workload

**Patrick Rusch**
Senior Consultant / IT Instructor

# Workload Types in Kubernetes

# Kubernetes Workload Types

Deployments

ReplicaSets

StatefulSets

DaemonSets

Jobs

CronJobs

# Choosing the Right Workload

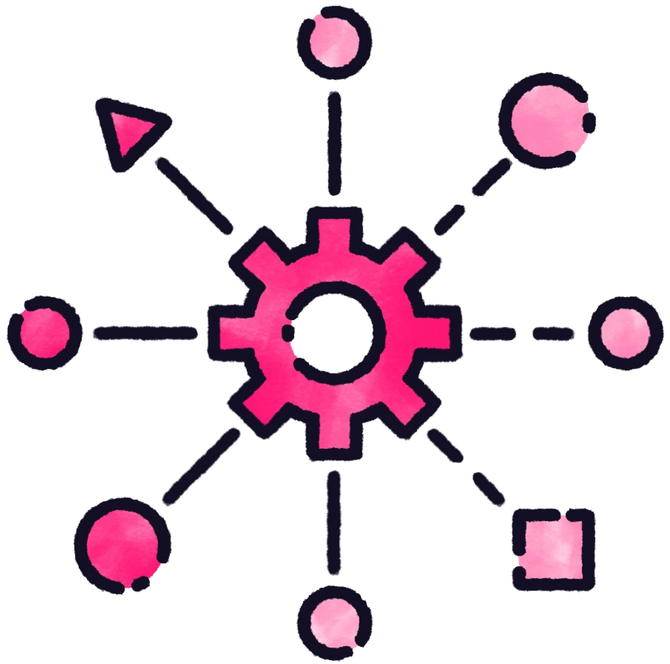
Ensure your application behave as expected

Scaling efficiently

Optimal management

# Deployments vs. ReplicaSets vs. StatefulSets

# What Are Deployments?

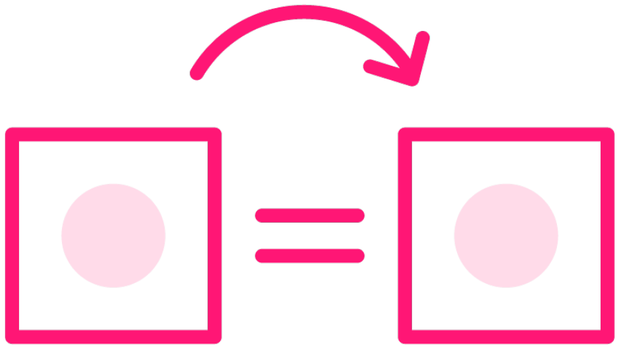


Stateless applications

Automatic scaling

Rolling updates and rollbacks

# What Are ReplicaSets?

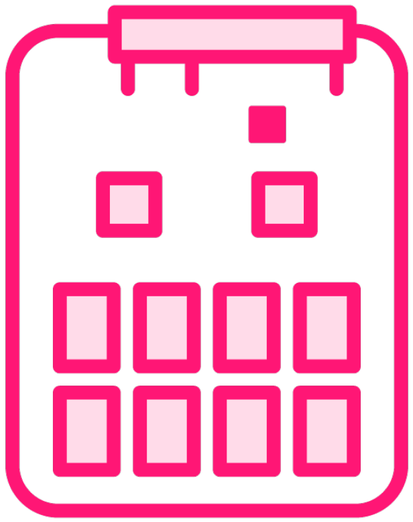


Ensure a set number of pods

No rolling updates or rollbacks

Often managed by Deployments

# What Are StatefulSets?

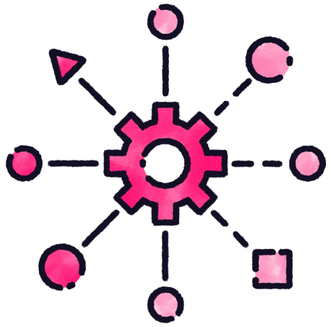


Stateful applications

Stable network identities

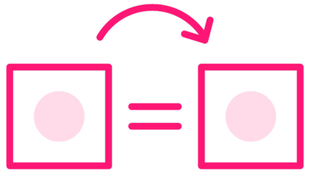
Persistent storage per pod

# When to Use What



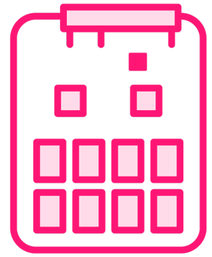
## Deployments

Stateless applications, which need to be scaled and updated automatically



## ReplicaSets

Applications that need a set number of instances

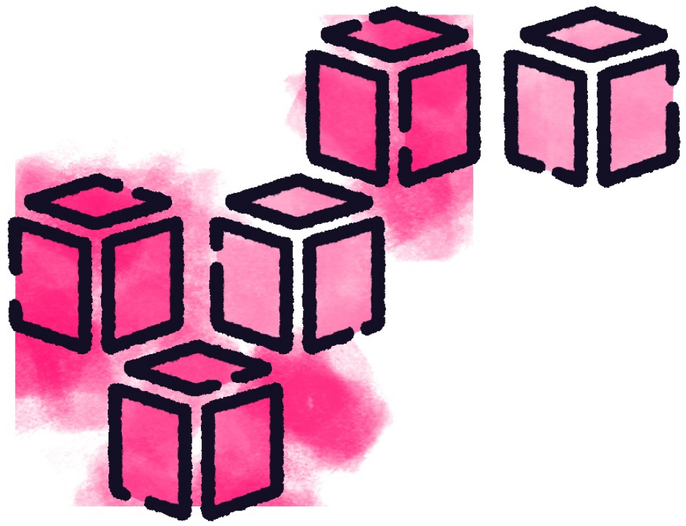


## StatefulSets

Applications that require persistent storage

# Background Services with DaemonSets

# What Are DaemonSets?



Ensure one pod per node

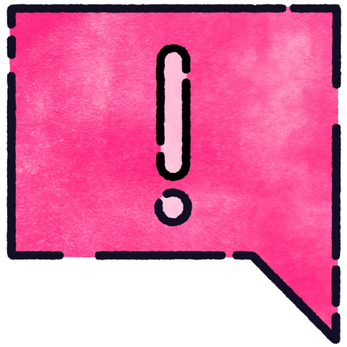
Ideal for background services

Schedule pods automatically on new nodes

Automatic updates and deletion

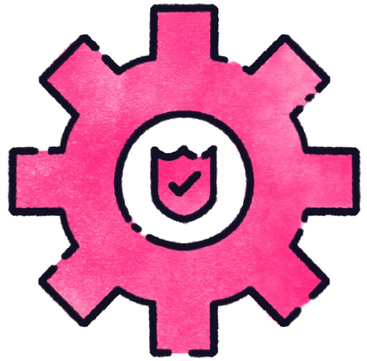
# Common Use Cases for DaemonSets


Log aggregation



Monitoring

Network management



Security agents

# When to Use and Not Use DaemonSets

Use for background services on every node

Don’t use if the service doesn’t need to run on every node or if scalability is required

# Scheduled Jobs with CronJobs

# What Is a CronJob?

CronJob schedules Jobs at specific times

Ideal for recurring tasks like backups or report generation

Runs periodically according to a cron schedule

# How CronJobs Work

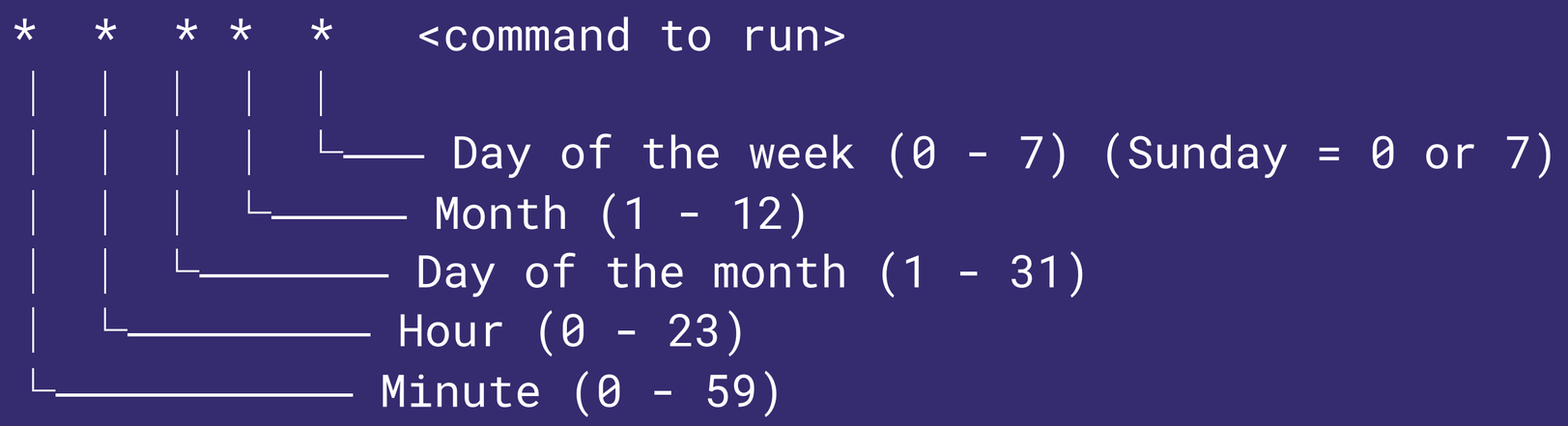
CronJob defines the schedule

CronJob triggers jobs at the scheduled time

Job runs the defined task and completes

# CronJob Schedule Format

CronJobs are scheduled using a format similar to traditional cron



# CronJob Examples

## Every day at midnight

```
0 0 * * *
```

## Every Monday at 9 AM

```
0 9 * * 1
```

# Common Use Cases for CronJobs

- Database backups
- Log rotation
- Report generation
- System cleanup

# How to Create a CronJob

## An example of a CronJob definition in Kubernetes

my-cronjob.yaml

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: my-cronjob
spec:
  schedule: "0 0 * * *"  # This will run at midnight every day
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: my-container
            image: my-image
          restartPolicy: OnFailure
```

# When to Use and Not Use CronJobs

Use for scheduled tasks, like backups and cleanups

Don’t use for tasks that need to run on-demand or scale dynamically

# One-time Tasks with Jobs

# What Is a Job?

Runs a task once until completion

Ideal for one-time tasks like database migrations or backups

Executes and exits upon completion

# How Jobs Work

- Defines the task to run
- Runs until completion, then marks as done
- Can retry in case of failure, depending on configuration

# When to Use a Job

- For one-time tasks that run to completion
- Suitable for batch processing, migrations, and backups
- Not intended for recurring tasks

# How to Create a Job

## An example of a Job definition in Kubernetes

my-job.yaml

```
apiVersion: batch/v1
kind: Job
metadata:
  name: my-job
spec:
  template:
    spec:
      containers:
      - name: my-container
        image: my-image
      restartPolicy: Never
```

# Controlling Retries and Concurrency

- Control retries with backoffLimit
- Set number of completions required with completions
- Manage concurrency with parallelism

# Demo: Pick the Right Controller and Deploy an App