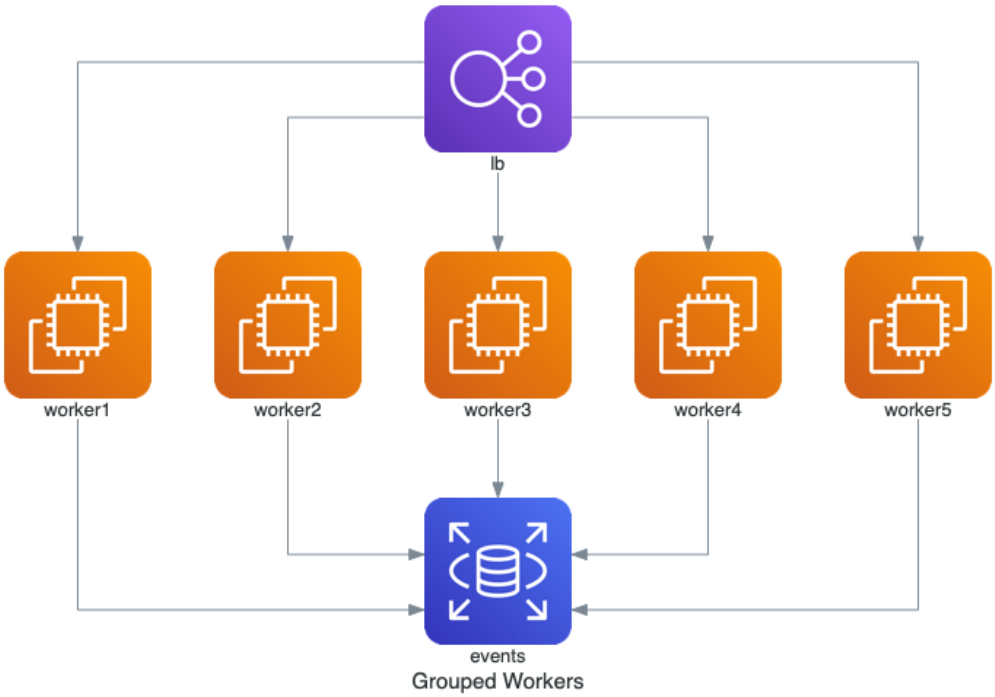
Examples

Here are some more examples.

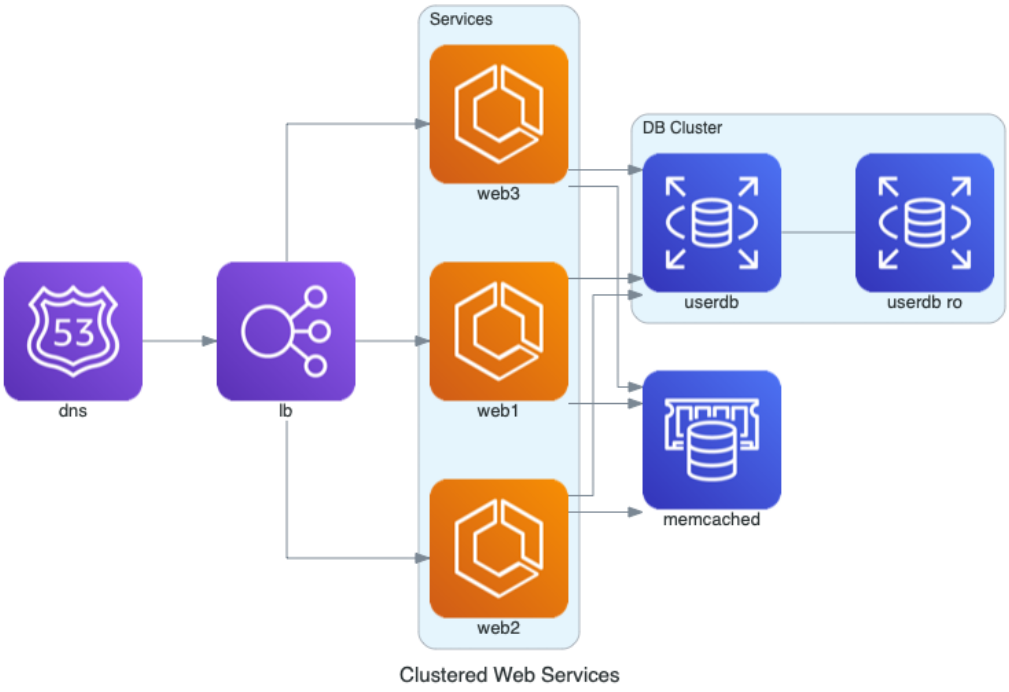
Grouped Workers on AWS

```
from diagrams import Diagram
from diagrams.aws.compute import EC2
from diagrams.aws.database import RDS
from diagrams.aws.network import ELB

with Diagram("Grouped Workers", show=False, direction="TB"):
    ELB("lb") >> [EC2("worker1"),
                  EC2("worker2"),
                  EC2("worker3"),
                  EC2("worker4"),
                  EC2("worker5")] >> RDS("events")
```



Clustered Web Services

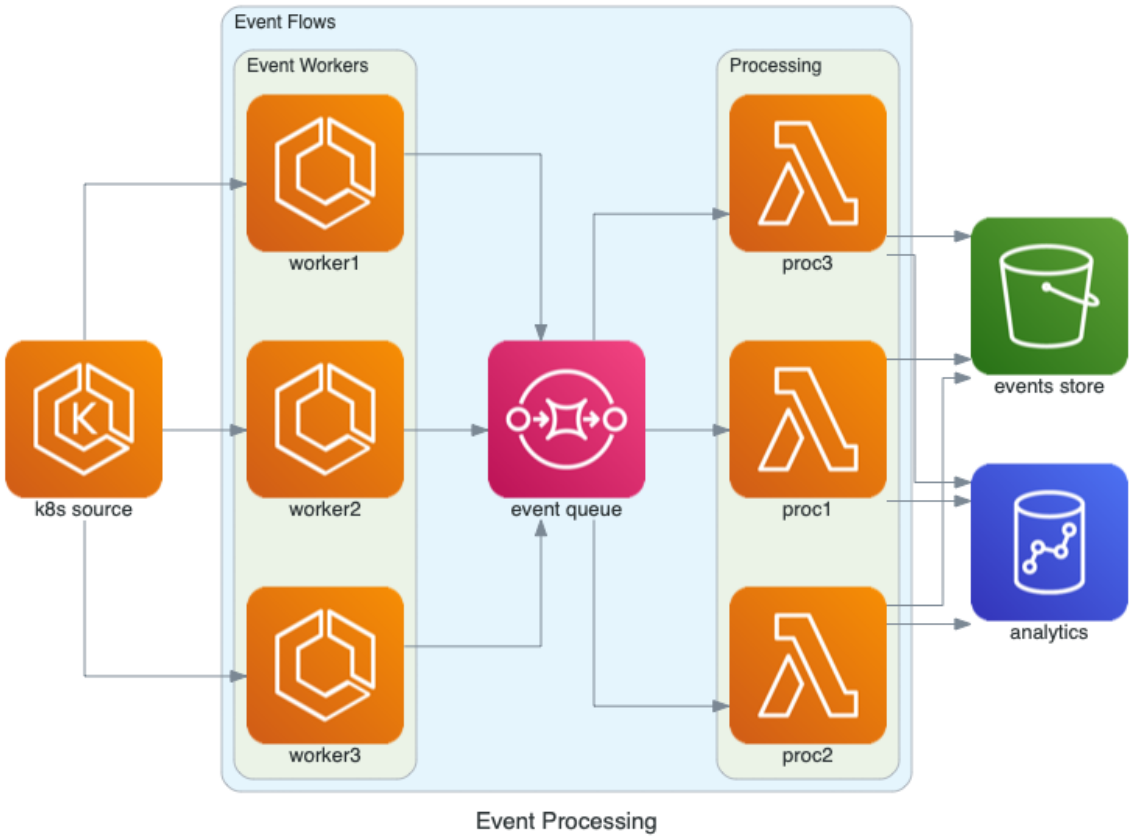
```
from diagrams import Cluster, Diagram
from diagrams.aws.compute import ECS
from diagrams.aws.database import ElastiCache, RDS
from diagrams.aws.network import ELB
from diagrams.aws.network import Route53

with Diagram("Clustered Web Services", show=False):
    dns = Route53("dns")
    lb = ELB("lb")

    with Cluster("Services"):
        svc_group = [ECS("web1"),
                     ECS("web2"),
                     ECS("web3")]

    with Cluster("DB Cluster"):
        db_primary = RDS("userdb")
        db_primary - [RDS("userdb ro")]

    memcached = ElastiCache("memcached")

    dns >> lb >> svc_group
    svc_group >> db_primary
    svc_group >> memcached
```



Event Processing on AWS

```
from diagrams import Cluster, Diagram
from diagrams.aws.compute import ECS, EKS, Lambda
from diagrams.aws.database import Redshift
from diagrams.aws.integration import SQS
from diagrams.aws.storage import S3

with Diagram("Event Processing", show=False):
    source = EKS("k8s source")

    with Cluster("Event Flows"):
        with Cluster("Event Workers"):
            workers = [ECS("worker1"),
                       ECS("worker2"),
                       ECS("worker3")]

        queue = SQS("event queue")

        with Cluster("Processing"):
            handlers = [Lambda("proc1"),
                        Lambda("proc2"),
                        Lambda("proc3")]

    store = S3("events store")
    dw = Redshift("analytics")
```

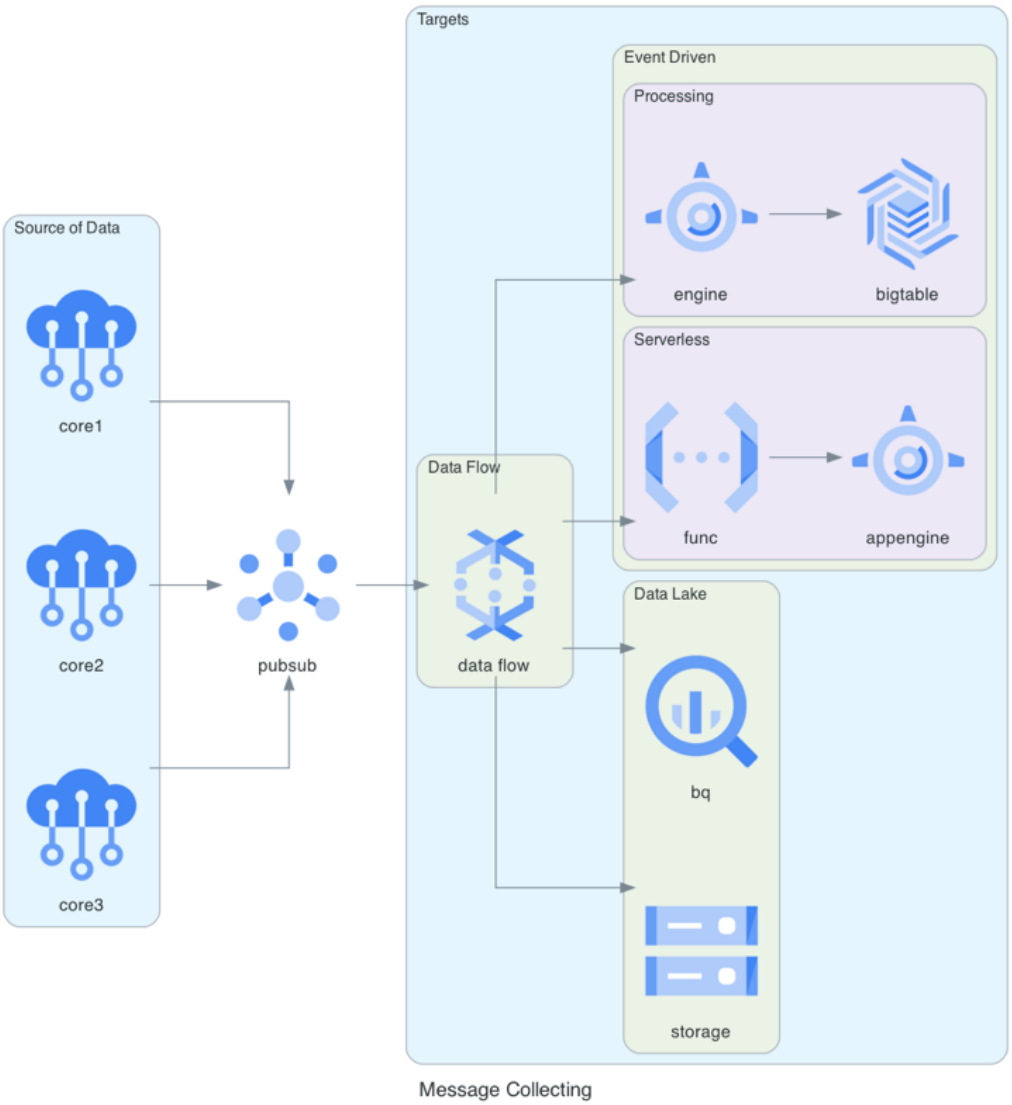
```
    source >> workers >> queue >> handlers
    handlers >> store
    handlers >> dw
```



Message Collecting System on GCP

```
from diagrams import Cluster, Diagram
from diagrams.gcp.analytics import BigQuery, Dataflow, PubSub
from diagrams.gcp.compute import AppEngine, Functions
from diagrams.gcp.database import BigTable
from diagrams.gcp.iot import IotCore
from diagrams.gcp.storage import GCS

with Diagram("Message Collecting", show=False):
    pubsub = PubSub("pubsub")

    with Cluster("Source of Data"):
        [IotCore("core1"),
         IotCore("core2"),
         IotCore("core3")] >> pubsub
```

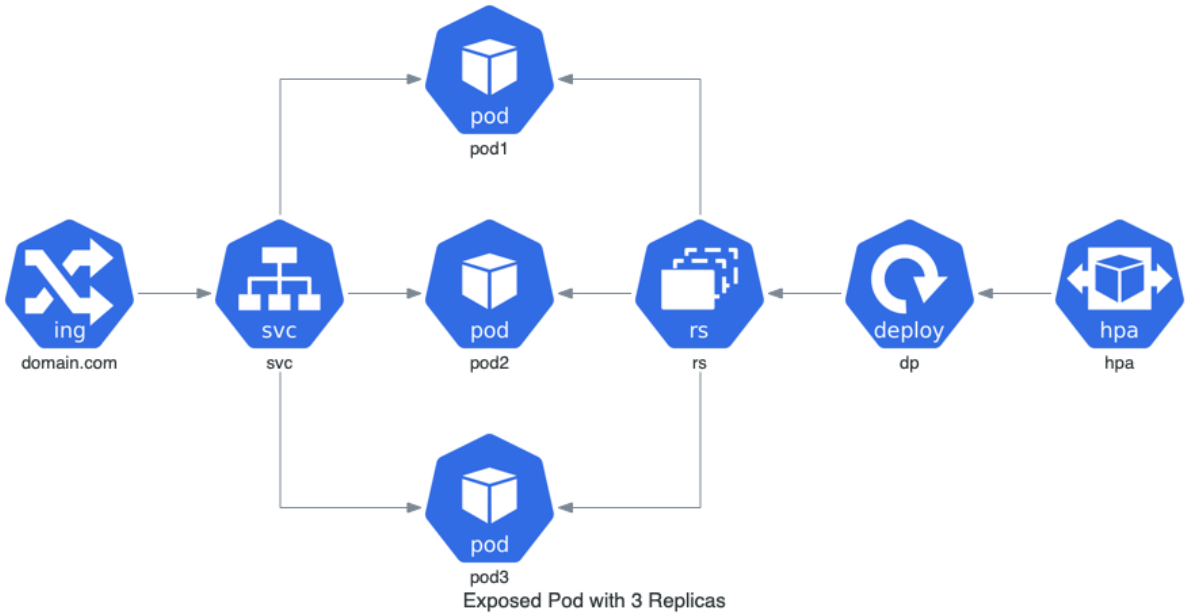
```
with Cluster("Targets"):
    with Cluster("Data Flow"):
        flow = Dataflow("data flow")

    with Cluster("Data Lake"):
        flow >> [BigQuery("bq"),
                 GCS("storage")]

    with Cluster("Event Driven"):
        with Cluster("Processing"):
            flow >> AppEngine("engine") >> BigTable("bigtable")

        with Cluster("Serverless"):
            flow >> Functions("func") >> AppEngine("appengine")

pubsub >> flow
```



Exposed Pod with 3 Replicas on Kubernetes

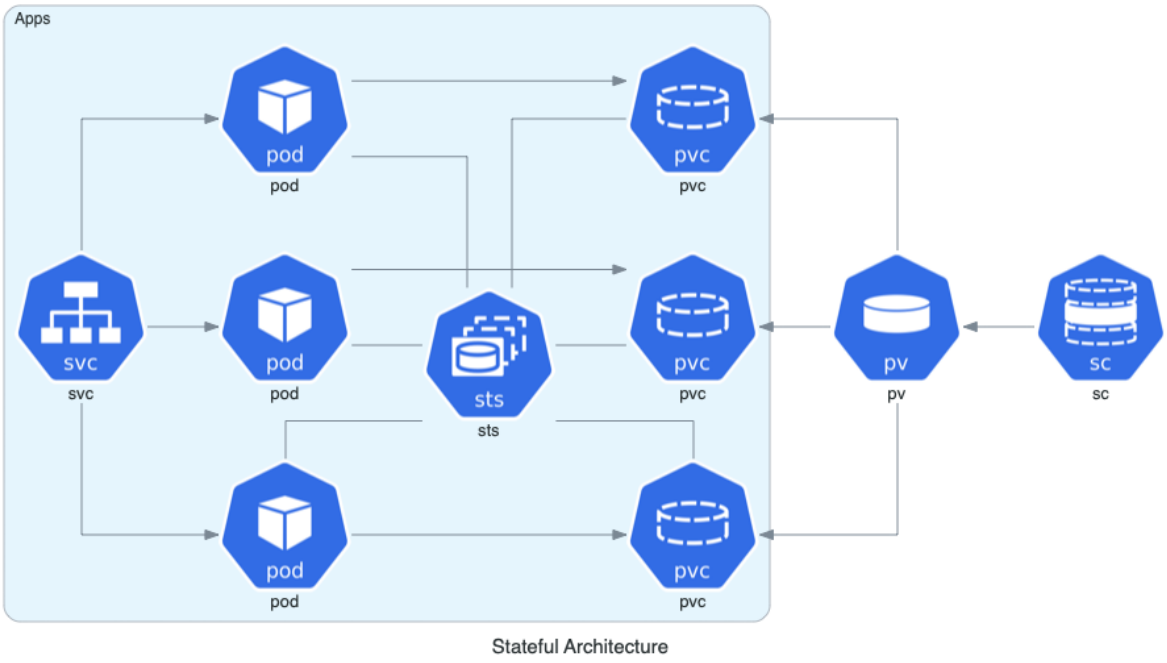
```
from diagrams import Diagram
from diagrams.k8s.clusterconfig import HPA
from diagrams.k8s.compute import Deployment, Pod, ReplicaSet
from diagrams.k8s.network import Ingress, Service

with Diagram("Exposed Pod with 3 Replicas", show=False):
    net = Ingress("domain.com") >> Service("svc")
    net >> [Pod("pod1"),
```

```
            Pod("pod2"),
            Pod("pod3")] << ReplicaSet("rs") << Deployment("dp") << HPA("hpa")
```



Stateful Architecture on Kubernetes

```
from diagrams import Cluster, Diagram
from diagrams.k8s.compute import Pod, StatefulSet
from diagrams.k8s.network import Service
from diagrams.k8s.storage import PV, PVC, StorageClass

with Diagram("Stateful Architecture", show=False):
    with Cluster("Apps"):
        svc = Service("svc")
        sts = StatefulSet("sts")

        apps = []
        for _ in range(3):
            pod = Pod("pod")
            pvc = PVC("pvc")
            pod - sts - pvc
            apps.append(svc >> pod >> pvc)

    apps << PV("pv") << StorageClass("sc")
```

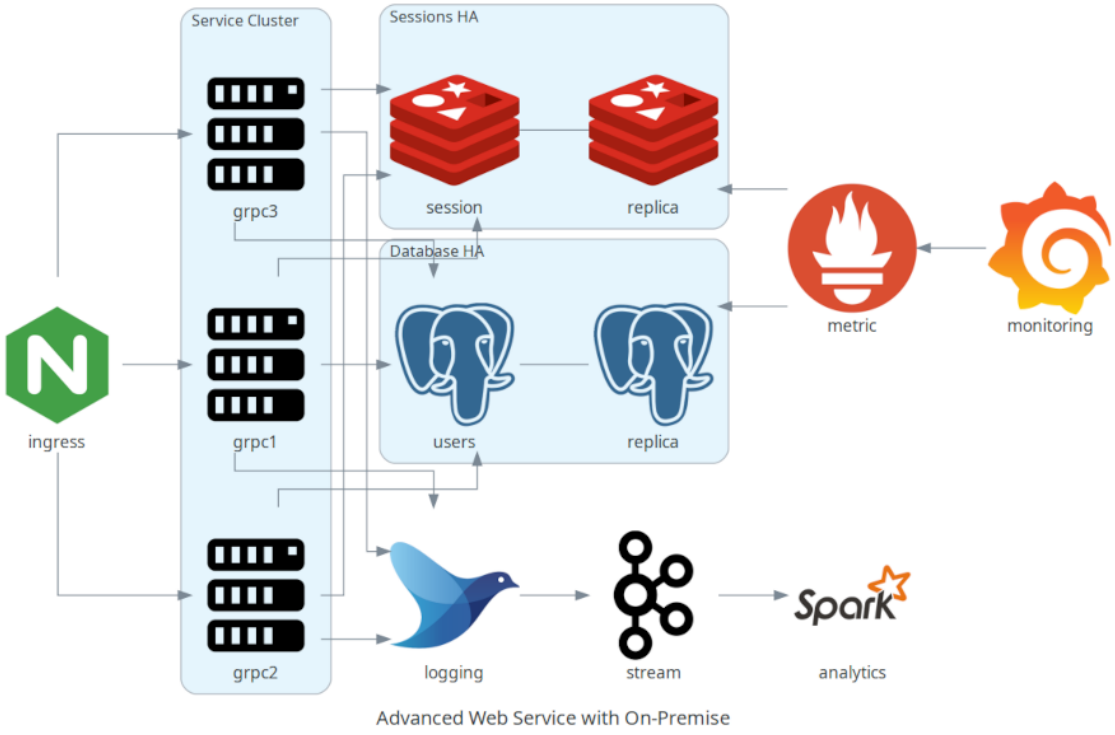


Stateful Architecture

Advanced Web Service with On-Premises

```
from diagrams import Cluster, Diagram
from diagrams.onprem.analytics import Spark
from diagrams.onprem.compute import Server
from diagrams.onprem.database import PostgreSQL
from diagrams.onprem.inmemory import Redis
from diagrams.onprem.aggregator import Fluentd
from diagrams.onprem.monitoring import Grafana, Prometheus
from diagrams.onprem.network import Nginx
from diagrams.onprem.queue import Kafka

with Diagram("Advanced Web Service with On-Premises", show=False):
    ingress = Nginx("ingress")

    metrics = Prometheus("metric")
    metrics << Grafana("monitoring")

    with Cluster("Service Cluster"):
        grpcsvc = [
            Server("grpc1"),
            Server("grpc2"),
            Server("grpc3")]

    with Cluster("Sessions HA"):
        primary = Redis("session")
        primary - Redis("replica") << metrics
        grpcsvc >> primary
```

```
    with Cluster("Database HA"):
        primary = PostgreSQL("users")
        primary - PostgreSQL("replica") << metrics
        grpcsvc >> primary

    aggregator = Fluentd("logging")
    aggregator >> Kafka("stream") >> Spark("analytics")

    ingress >> grpcsvc >> aggregator
```

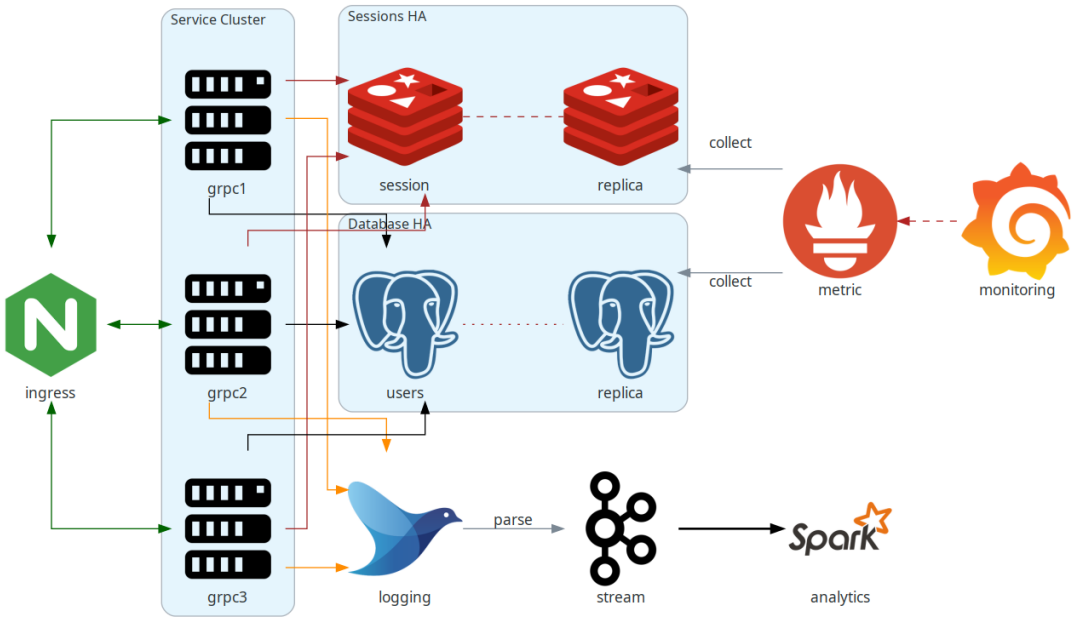


Advanced Web Service with On-Premise

Advanced Web Service with On-Premises (with colors and labels)

```
from diagrams import Cluster, Diagram, Edge
from diagrams.onprem.analytics import Spark
from diagrams.onprem.compute import Server
from diagrams.onprem.database import PostgreSQL
from diagrams.onprem.inmemory import Redis
from diagrams.onprem.aggregator import Fluentd
from diagrams.onprem.monitoring import Grafana, Prometheus
from diagrams.onprem.network import Nginx
from diagrams.onprem.queue import Kafka

with Diagram(name="Advanced Web Service with On-Premises (colored)", show=Fals
```

```
ingress = Nginx("ingress")

metrics = Prometheus("metric")
metrics << Edge(color="firebrick", style="dashed") << Grafana("monitoring"

with Cluster("Service Cluster"):
    grpcsvc = [
        Server("grpc1"),
        Server("grpc2"),
        Server("grpc3")]

with Cluster("Sessions HA"):
    primary = Redis("session")
    primary - Edge(color="brown", style="dashed") - Redis("replica") << Ed
    grpcsvc >> Edge(color="brown") >> primary

with Cluster("Database HA"):
    primary = PostgreSQL("users")
    primary - Edge(color="brown", style="dotted") - PostgreSQL("replica")
    grpcsvc >> Edge(color="black") >> primary

aggregator = Fluentd("logging")
aggregator >> Edge(label="parse") >> Kafka("stream") >> Edge(color="black"

ingress >> Edge(color="darkgreen") << grpcsvc >> Edge(color="darkorange")
```

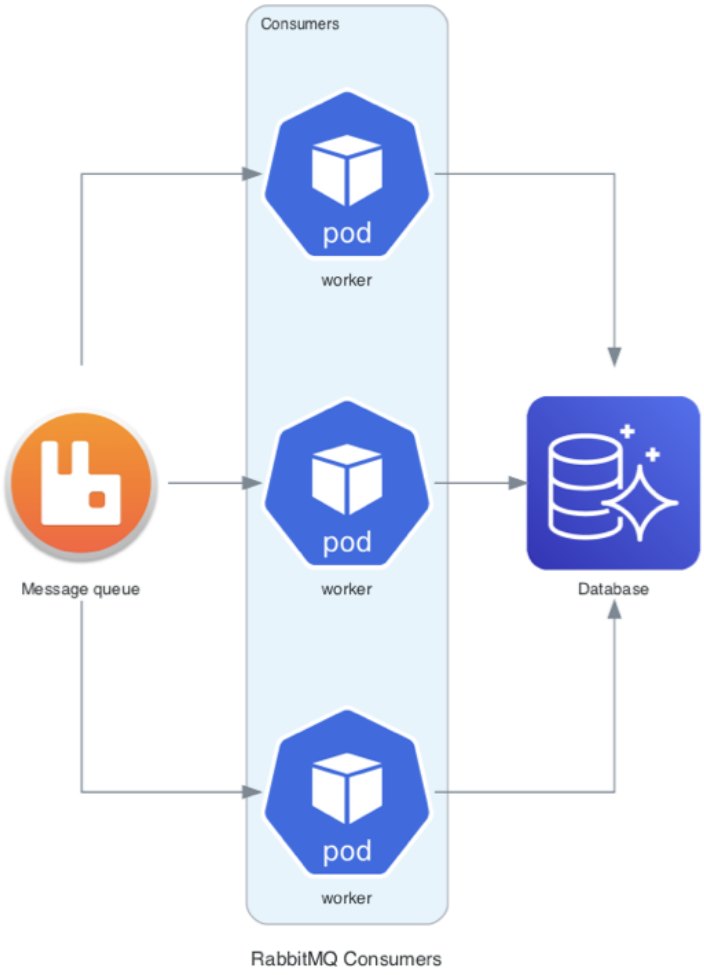


Advanced Web Service with On-Premise (colored)

RabbitMQ Consumers with Custom Nodes

```
from urllib.request import urlretrieve

from diagrams import Cluster, Diagram
from diagrams.aws.database import Aurora
from diagrams.custom import Custom
from diagrams.k8s.compute import Pod

# Download an image to be used into a Custom Node class
rabbitmq_url = "https://jpadilla.github.io/rabbitmqapp/assets/img/icon.png"
rabbitmq_icon = "rabbitmq.png"
urlretrieve(rabbitmq_url, rabbitmq_icon)

with Diagram("Broker Consumers", show=False):
    with Cluster("Consumers"):
        consumers = [
            Pod("worker"),
            Pod("worker"),
            Pod("worker")]

    queue = Custom("Message queue", rabbitmq_icon)
```

```
queue >> consumers >> Aurora("Database")
```



Last updated on 2/22/2025

← INSTALLATION

DIAGRAMS →