Apache Airflow with AWS Fargate

Deependra Shekhawat

Agenda

- Introduction - Apache Airflow
- Evolving Airflow installations
- Highly Scalable and Highly Available Airflow installation - AWS Fargate
- Lessons Learned

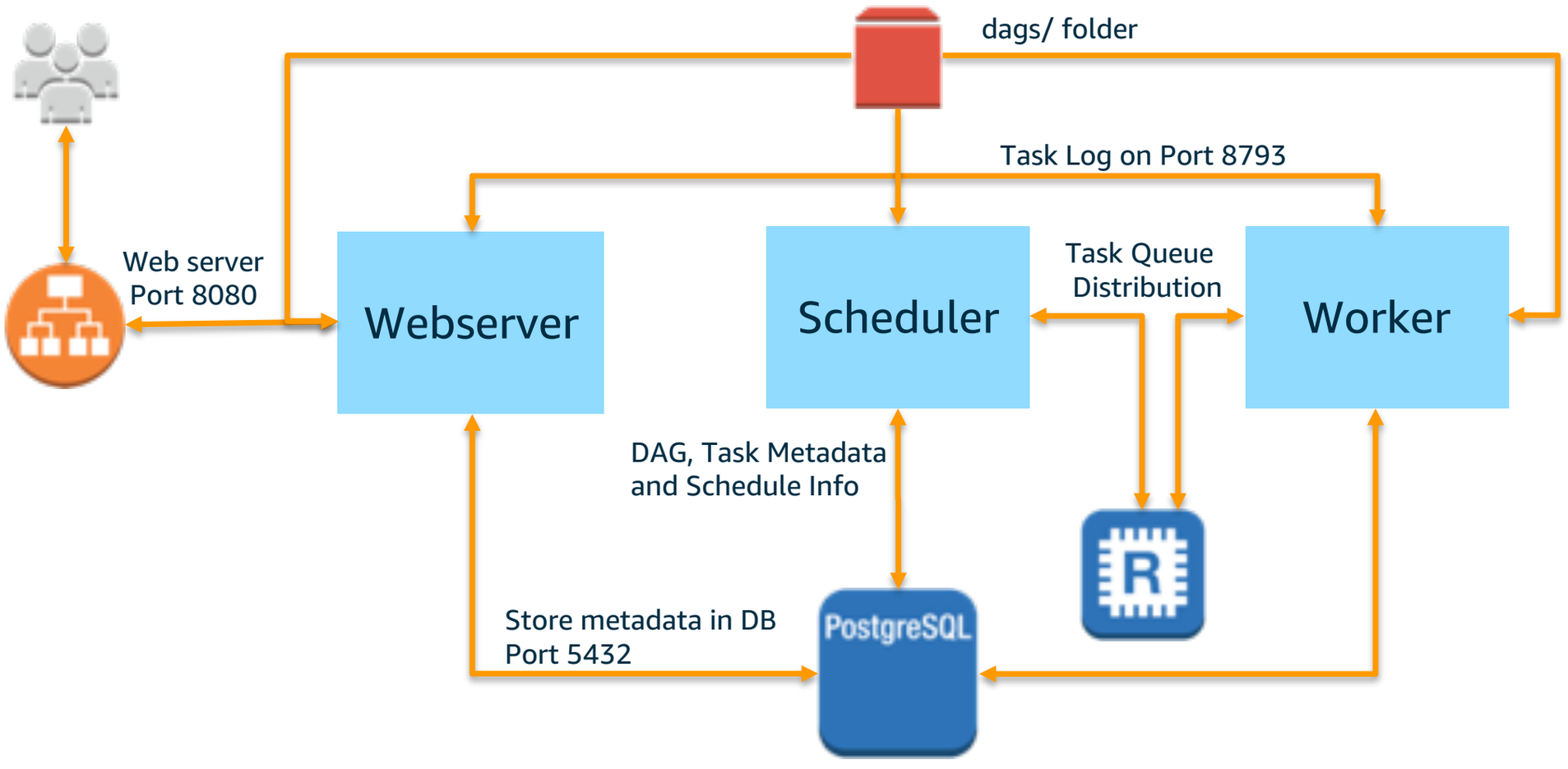
Apache Airflow - Introduction

- Platform to programmatically author, schedule and monitor workflows.
- Data processing pipelines as code (Python)
- Key concepts - DAG, DAG Run, Operator, Task, Task Instance, **Executor**
- Key Components - Web Server, Scheduler, Worker, Task Queue

Evolving Airflow Installations on AWS

- Single EC2 instance - Use SequentialExecutor (1 worker)
- Single EC2 instance - Use LocalExecutor (Multiple workers, using IPC queues)
- N Webservers, 1 Scheduler + Worker - Use LocalExecutor
- N Webservers, 1 Scheduler, N workers - Use CeleryExecutor (Fully distributed task queues)
- Run Airflow inside Docker container

Apache Airflow - Architecture



How do I get Airflow on Fargate?

Launching Airflow with AWS Fargate

- Amazon Elastic Container Service (Amazon ECS) - Container management service for Docker containers

- A compute engine for Amazon ECS that allows you to run containers without having to manage servers or clusters.

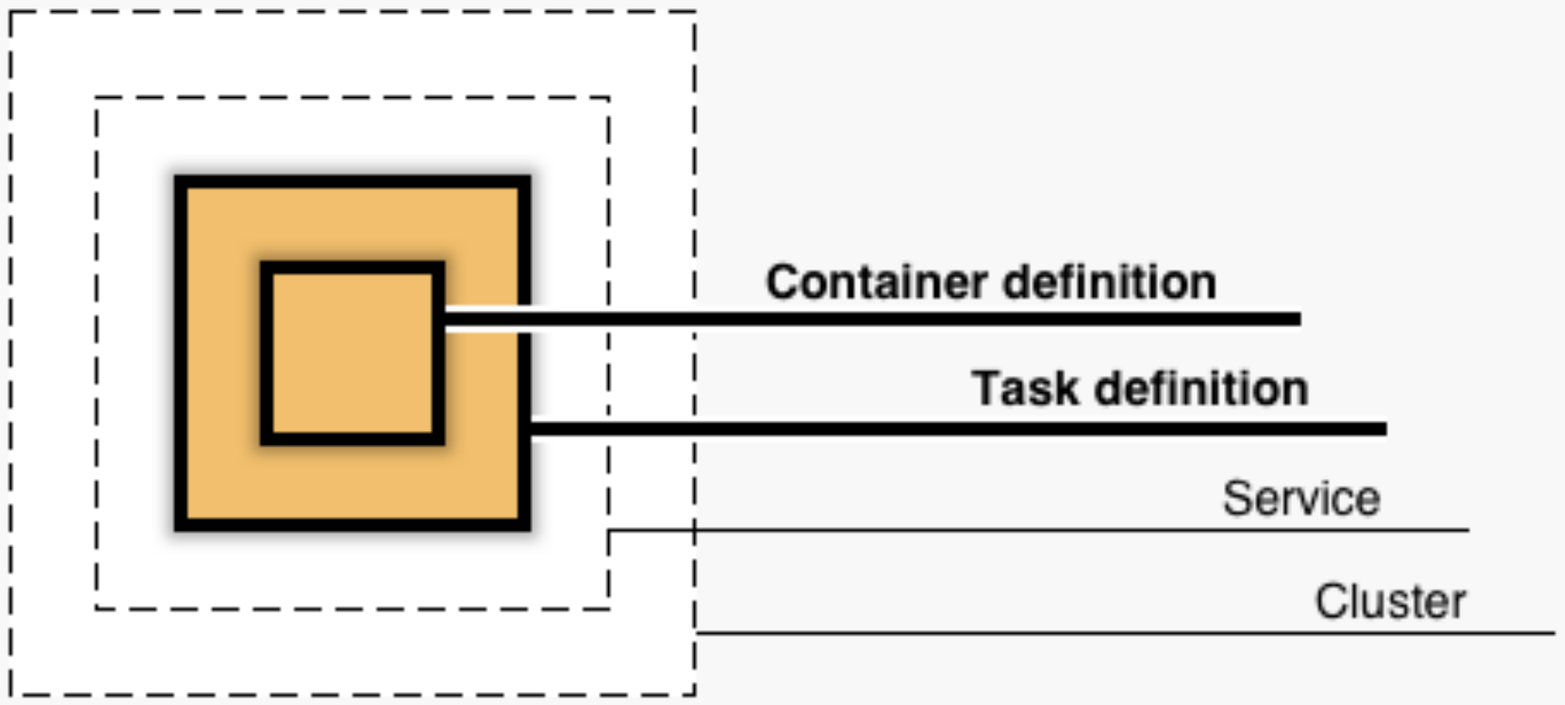
- Package application in containers (**DockerFile** / Docker Registry)

- Specify CPU and memory requirements (**Task Definition**)

- Define Networking and IAM policies

- Define Application as **Service** Launch!!

Elastic Container Service - Constructs



step 0 - Setup the dependencies

- Define VPC, Subnets, Security Groups, Route Tables, NACLs
- Setup dependencies -
  - Amazon RDS PostgreSQL (Airflow Metadata)
  - Amazon ElastiCache (Redis as Message Broker for Celery)
  - AWS Systems Manager Parameter Store (DB credentials, Fernet Key, Redis URL)
  - Application Load balancer (For Airflow Web server)

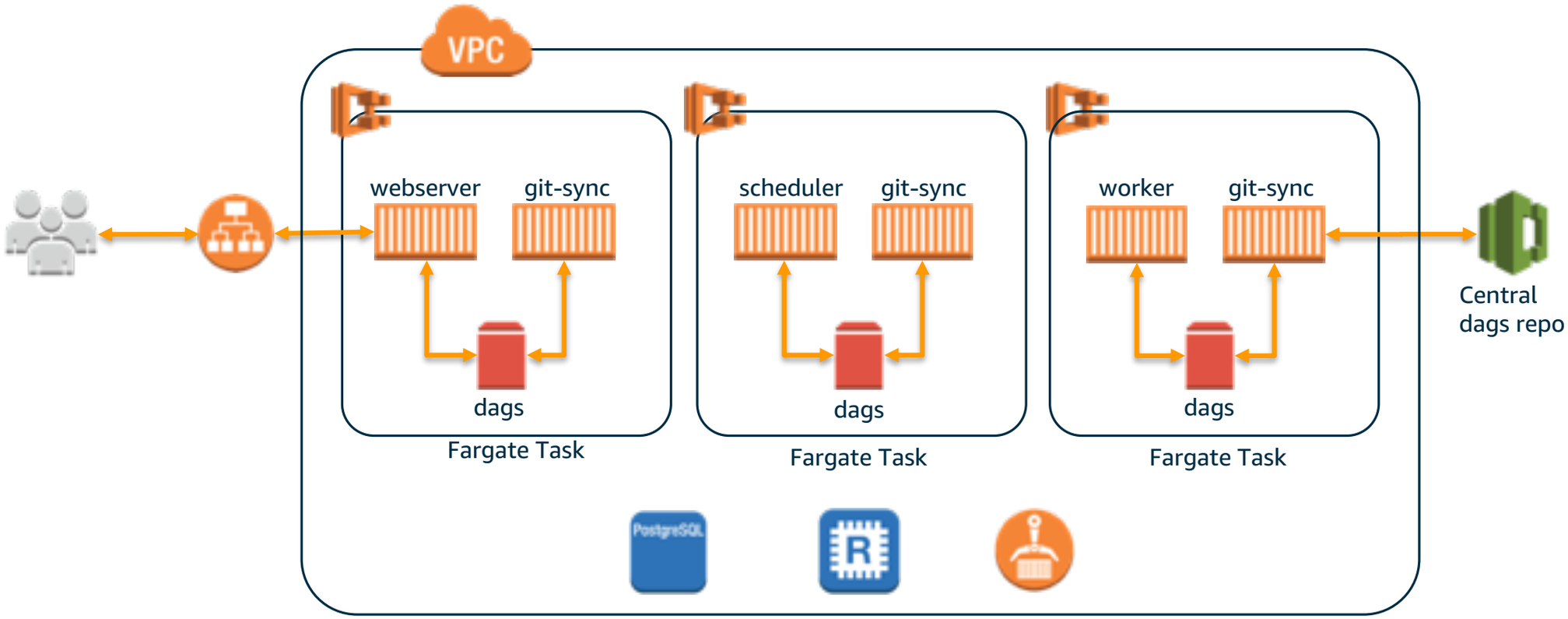
step 1 - Setup Fargate Tasks

- Prepare Docker Image for Airflow
  - https://github.com/jeevanullas/docker-airflow/tree/fargate
  - Includes support for Parameter Store
    - Includes airflow_local_settings to fix hostname <->IP mapping issue via hostname_callable config param
- Prepare Fargate Task Definitions
  - https://github.com/jeevanullas/docker-airflow/tree/fargate/ecstaskdef
  - webserver, scheduler and worker.
  - git-sync as sidecar container for dags/ folder

step 2 - Launch ECS Cluster and services

- Create ECS managed cluster
- Create ECS services within Cluster
  - webserver - Multiple tasks behind ALB for Load balancing (Use Autoscaling)
  - scheduler - Run only 1 task (REPLICA SchedulingStrategy) (Autohealing)
  - worker - Run several tasks (Use Autoscaling)

Airflow on Fargate



Airflow - Lessons Learned

- Use a managed database for the metadata
- Use a distributed task queue for your tasks
- Shared filesystem for your DAGs - Ideally managed via git
- Scale your workers based on meaningful metrics
    - Vertical scaling - Use Airflow Queues and different worker services
    - Horizontal scaling - Scale your workers based on tasks in the task queue (requires effort)
- Logs - Use persistent storage like S3 / CloudWatch logs
- CI / CD for your infrastructure and DAGs

Some Useful Links

- ETL best practices with Airflow documentation site - https://gtoonstra.github.io/etl-with-airflow/
- Built to Scale: Running Highly-Concurrent ETL with Apache Airflow - https://bostata.com/post/built-to-scale-running-highly-concurrent-etl-with-apache-airflow/
- Airflow with Docker - https://github.com/puckel/docker-airflow

Questions