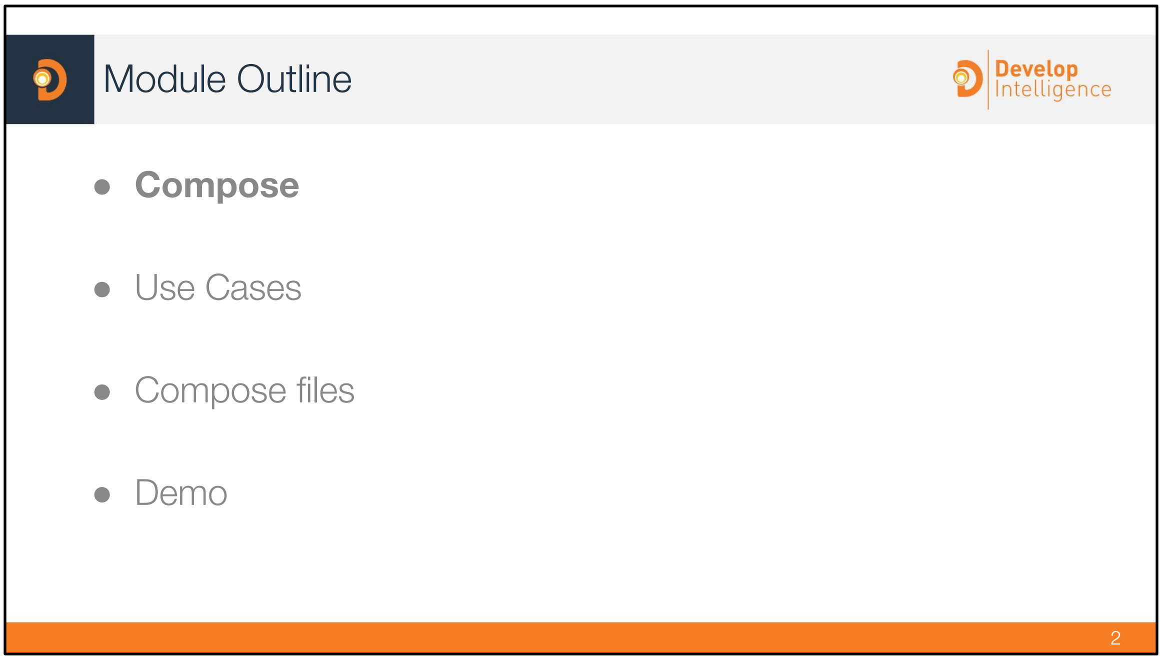
Docker Compose

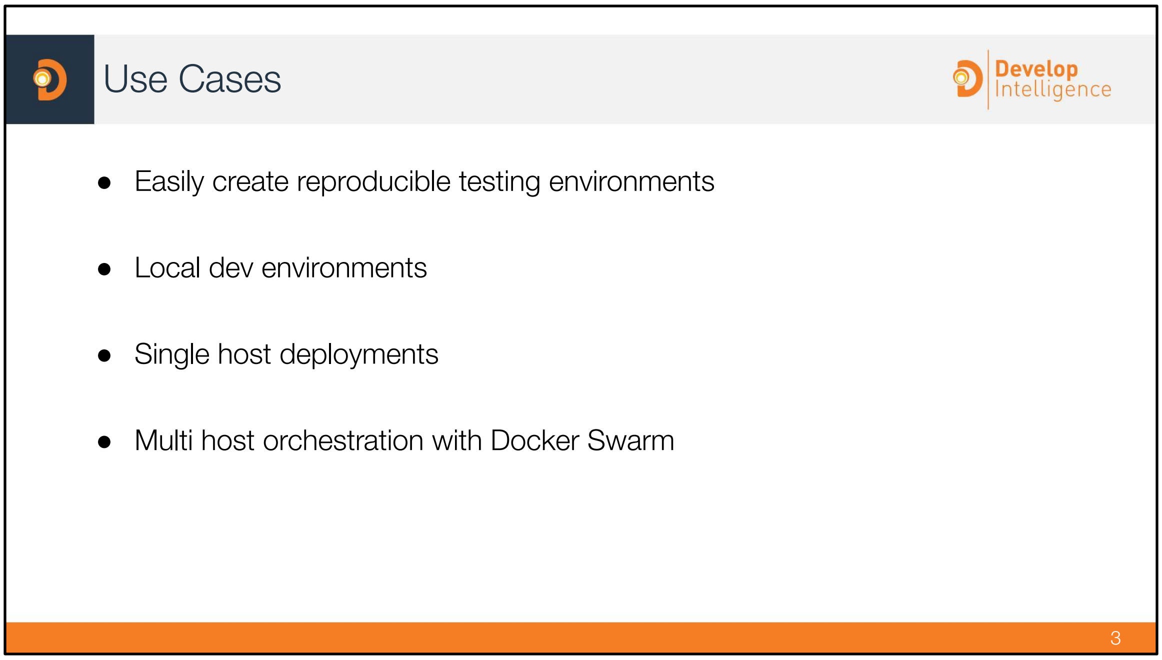
Develop Intelligence



Module Outline

Develop Intelligence

- **Compose**
- Use Cases
- Compose files
- Demo



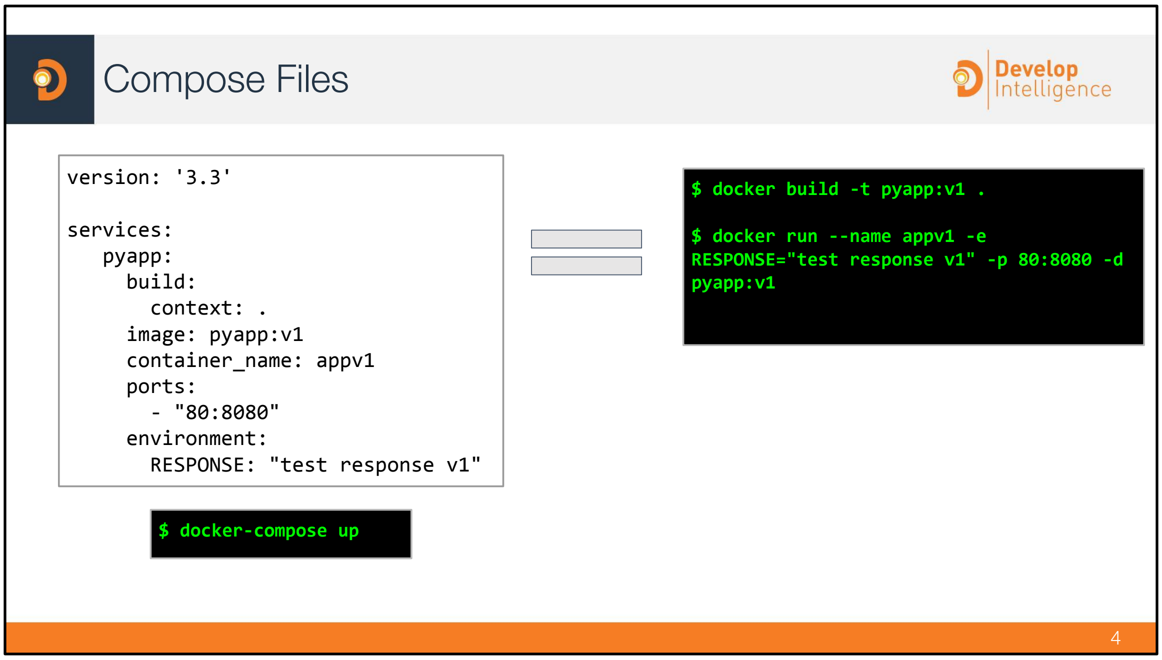
Use Cases

Develop Intelligence

- Easily create reproducible testing environments
- Local dev environments
- Single host deployments
- Multi host orchestration with Docker Swarm

3

Similar to Vagrant for creating and starting up dev environments. Or helm for deploying complex applications onto Kubernetes. A simple way to treat a multiple container application as a single unit.



Compose Files

```
version: '3.3'

services:
   pyapp:
     build:
       context: .
     image: pyapp:v1
     container_name: appv1
     ports:
       - "80:8080"
     environment:
       RESPONSE: "test response v1"
```

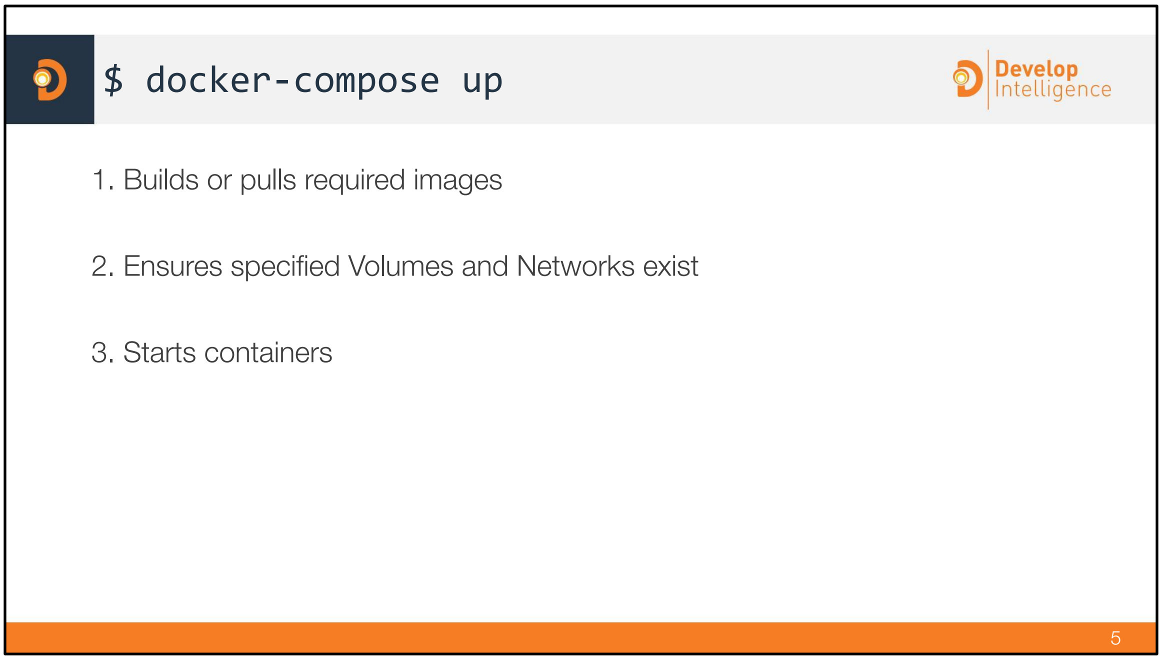
```
$ docker-compose up
```

```
$ docker build -t pyapp:v1 .

$ docker run --name appv1 -e
RESPONSE="test response v1" -p 80:8080 -d
pyapp:v1
```

4

With compose, you simply define a file that has all of the options specified for starting your container. This gives you an easy way to consistently run your application, and this compose file could be version controlled and shared easily.



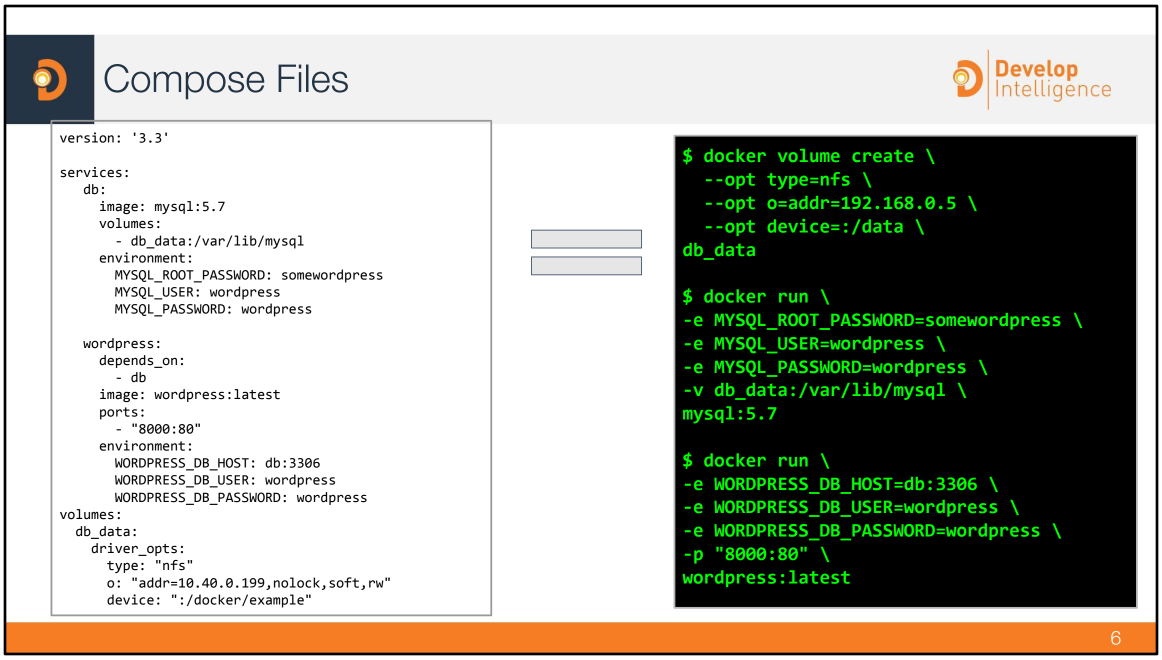
$ docker-compose up

Develop Intelligence

1. Builds or pulls required images
2. Ensures specified Volumes and Networks exist
3. Starts containers

5

This single command will read your compose file and do everything needed to run the application. This might include building the contain images, creating a network on the host for the containers to run in, creating volumes, and then finally starting the containers.



Compose Files

Develop Intelligence

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
volumes:
  db_data:
    driver_opts:
      type: "nfs"
      o: "addr=10.40.0.199,nolock,soft,rw"
      device: ":/docker/example"
```

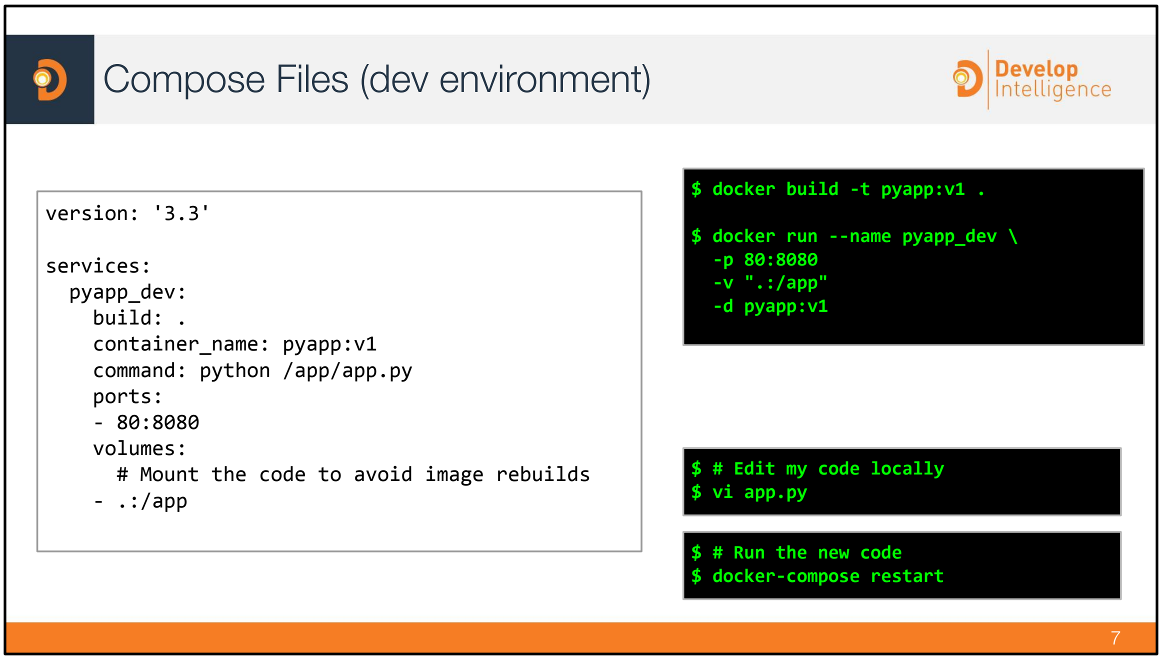
```
$ docker volume create \
  --opt type=nfs \
  --opt o=addr=192.168.0.5 \
  --opt device=:/data \
db_data

$ docker run \
-e MYSQL_ROOT_PASSWORD=somewordpress \
-e MYSQL_USER=wordpress \
-e MYSQL_PASSWORD=wordpress \
-v db_data:/var/lib/mysql \
mysql:5.7

$ docker run \
-e WORDPRESS_DB_HOST=db:3306 \
-e WORDPRESS_DB_USER=wordpress \
-e WORDPRESS_DB_PASSWORD=wordpress \
-p "8000:80" \
wordpress:latest
```

6

The real power comes in complex setups. One thing to note, when volumes are created by docker compose, they must be explicitly deleted with the '--volumes' flag on docker-compose down commands. This is so you can redeploy your application and have the data persist.



Compose Files (dev environment)

Develop Intelligence

```
version: '3.3'

services:
  pyapp_dev:
    build: .
    container_name: pyapp:v1
    command: python /app/app.py
    ports:
    - 80:8080
    volumes:
      # Mount the code to avoid image rebuilds
    - .:/app
```

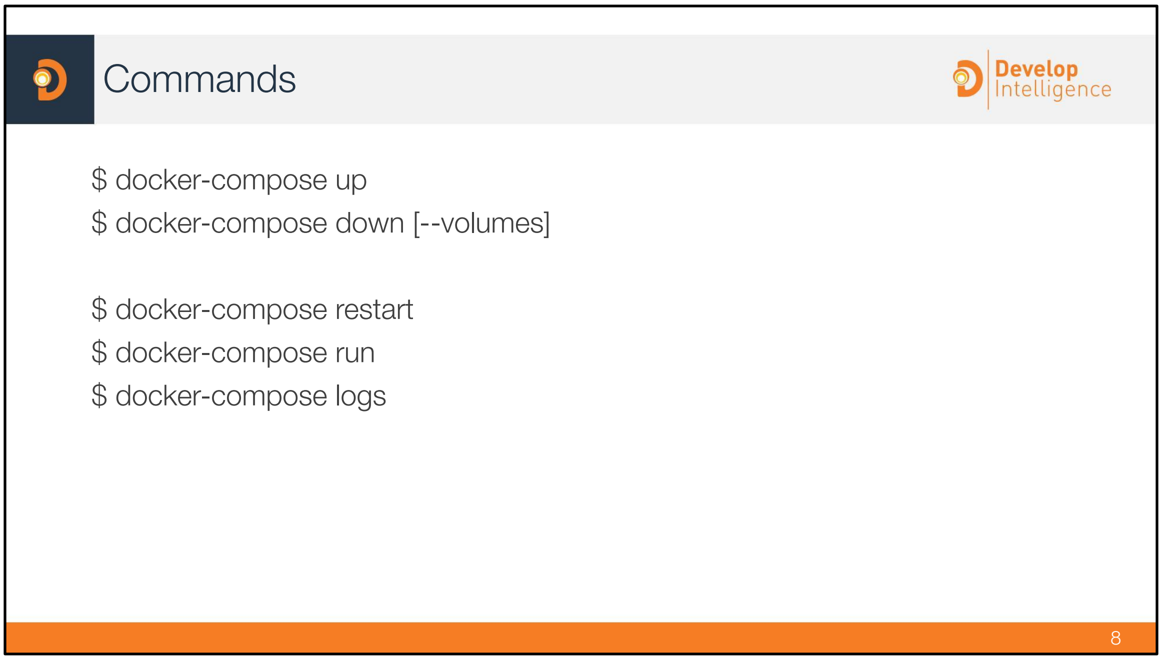
```
$ docker build -t pyapp:v1 .

$ docker run --name pyapp_dev \
  -p 80:8080
  -v ".:/app"
  -d pyapp:v1
```

```
$ # Edit my code locally
$ vi app.py
```

```
$ # Run the new code
$ docker-compose restart
```

7

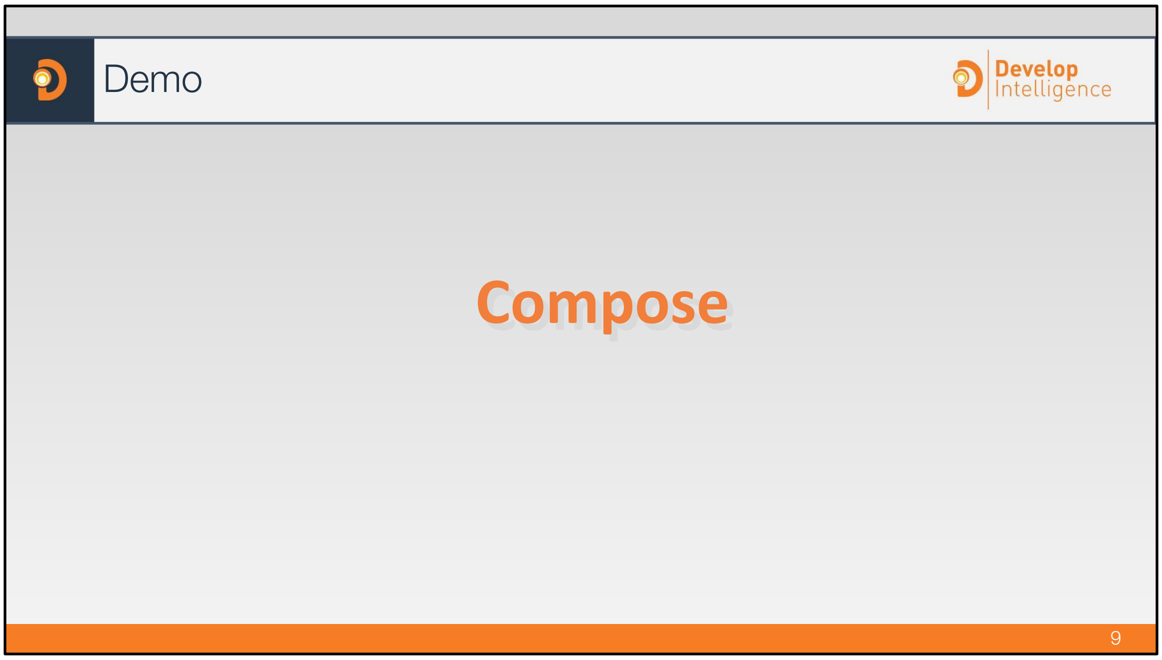


Commands

```
$ docker-compose up
$ docker-compose down [--volumes]

$ docker-compose restart
$ docker-compose run
$ docker-compose logs
```

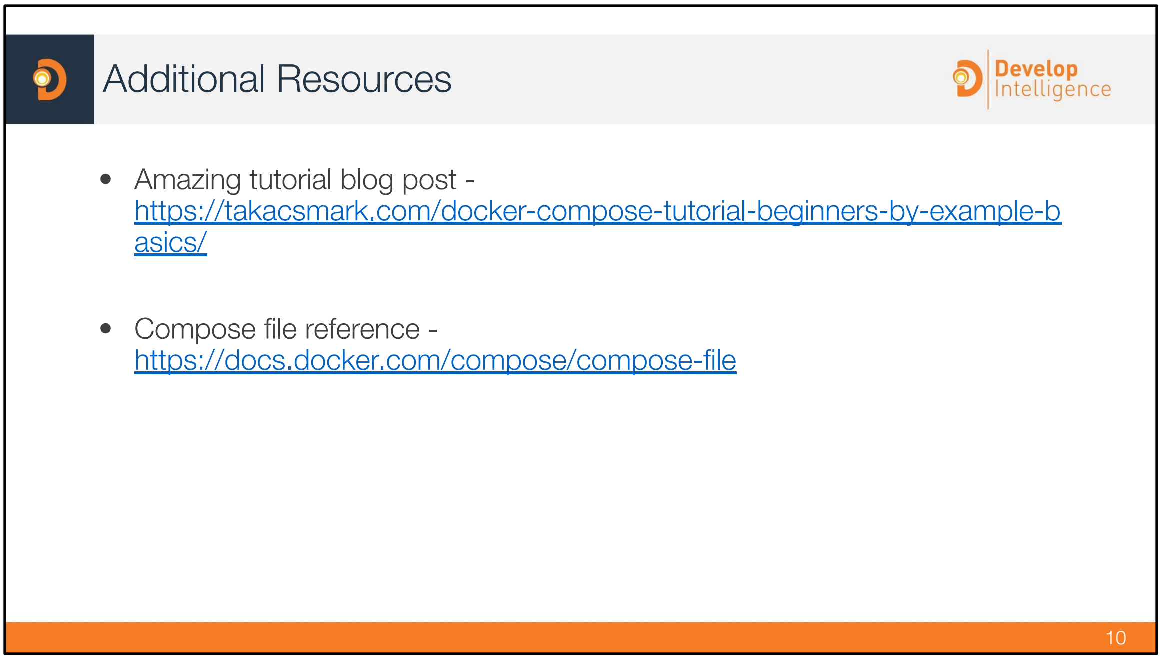
8



Demo

Develop Intelligence

Compose



Additional Resources

- Amazing tutorial blog post - https://takacsmark.com/docker-compose-tutorial-beginners-by-example-basics/
- Compose file reference - https://docs.docker.com/compose/compose-file

10