Docker Swarm Case Study

How Omilia started running containers in production

2hog.codes

2hog

We are a software consultancy and training team based in Athens, Greece.

Our work focuses on bringing **calmness** and increasing **productivity** of software teams

Most of the time we:

- **Containerize** applications
- Set up **Continuous Integration and Continuous Delivery**
- Instrument **error tracking and performance monitoring**

Paris Kasidiaris

@pariskasid

- Co-founder at SourceLair / stolos.io
- Creator of xterm.js
- Docker and Python lover

Antonis Kalipetis

@akalipetis

- Docker Captain and early user
- Python lover and developer
- Technology lead at SourceLair / stolos.io
- Docker training and consulting

Some context first

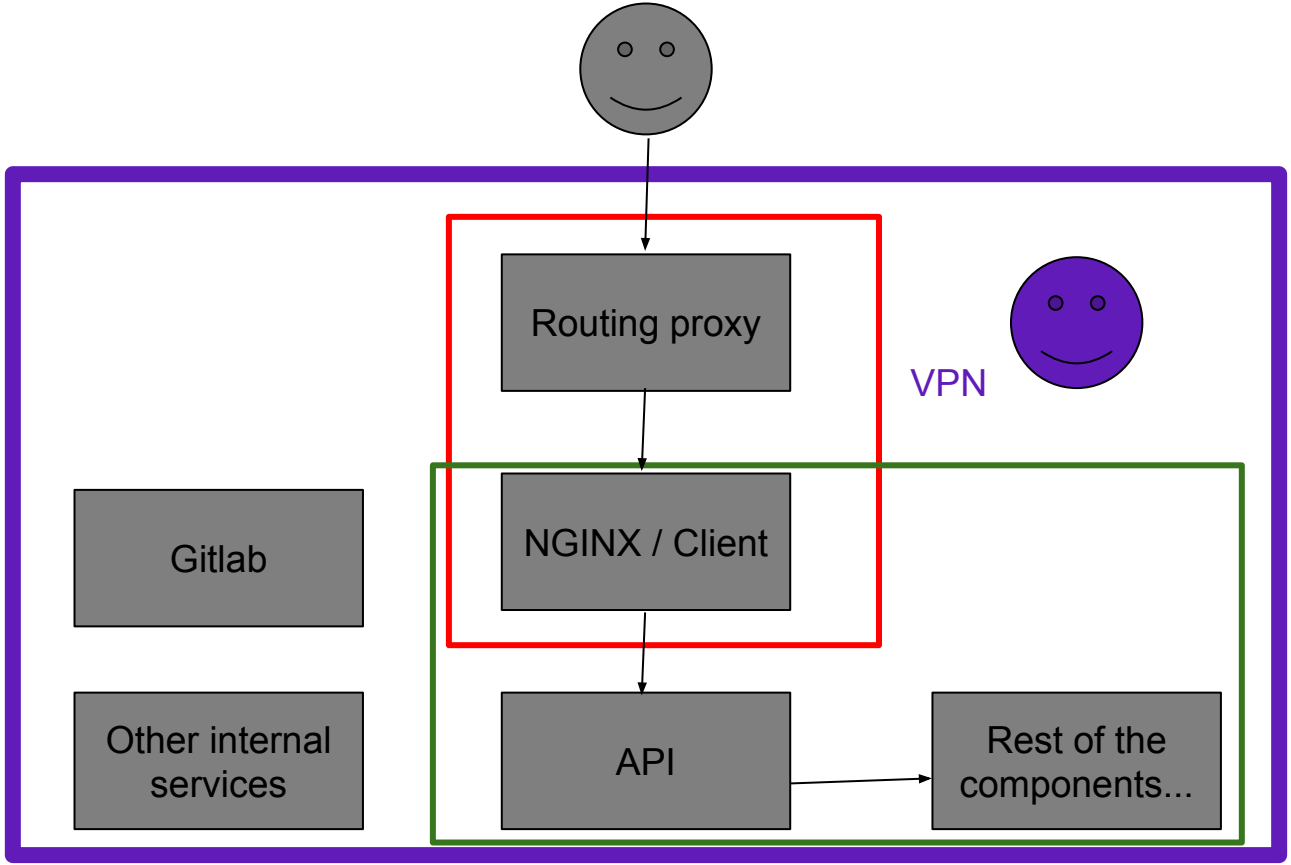
Architecture

Architecture

The components

- Single page app client, built with React, served by NGINX
- API server, built with Python Django
- Workers, built with Python Celery
- PostgreSQL database
- Redis cache + queue
- NLU API
- DiaManT conversation management API
- Machine Learning

Architecture



First challenge

Splitting the monolith

Moving to containers cannot just be a lift-and-shift operation. Things will need to change, at least a bit.

Continuous integration and delivery

Continuous integration and delivery

- From deployments per year, to deployments per day
- Verified and tested deployments
- Deployment history, allowing point in time rollbacks

Second challenge

From EmailOps to DevOps

Releasing early and repeatedly allows for finding, fixing and rolling back bugs fast. It also keeps stakeholders happy, not waiting.

What the heck is happening

What the heck is happening

- There are multiple containers running, in different servers
- You can’t *curl* your way to debugging
- There’s *black magic* happening

Third challenge

Getting yourself together, monitoring and understanding the infrastructure.

Releasing early and repeatedly allows for finding, fixing and rolling back bugs fast. It also keeps stakeholders happy, not waiting.

Solutions chosen

Orchestration

Docker Swarm

Docker Swarm is our orchestrator of choice and was a perfect fit, as it dead-easy to operate and secure by default.

Continuous integration

Gitlab CI

Gitlab CI was chosen, as it has Dockerized pipelines at its core, already has a Docker registry and Gitlab was already deployed as a code hosting tool.

Monitoring

Prometheus with Grafana

Prometheus pull model and the out-of-the-box integration with Docker Swarm made the choice easy. The Grafana public dashboards the cherry on the cake.

Current state

Current state

- 2 Docker Swarm clusters, 3-node and single-node
- Every public facing and new service, is deployed first with Docker
- Standard processes and stubs for testing, building and deploying services

The process

Testing

- All branches and MRs are tested automatically
- MRs do not get merged without tests passing
- Tests are run inside Docker, using Docker Compose

Staging Deployment

- Deployments are automatically handled with a branch strategy
- Staging deployment is identical with production

Production Deployment

- Production deployment is manually triggered
- The same process guarantees deployment success
- Separate environments allow for easier rollbacks

Some stats

2

Swarm clusters

56

G of RAM

40

Services running

2000+

Deployments

Thank you!