How to deploy Docker on AWS with no downtime

Dmitry Gusev

AnjLab TechTalks
Vladimir, Russia
2017-02-03

Agenda

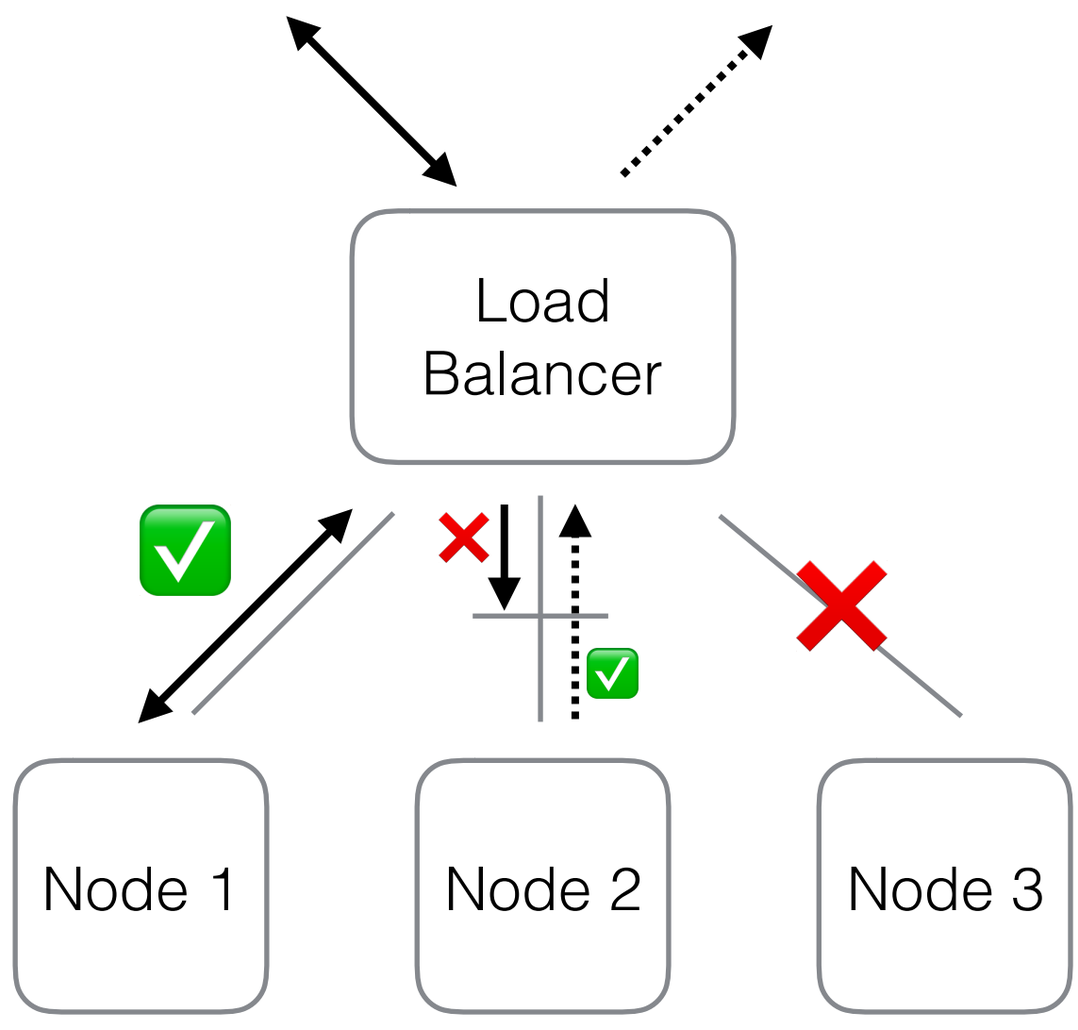
- Zero-downtime deployment
- Docker deployment options for AWS
- Proposed solution
- Demo

Zero-downtime Deployment

- [Stateless] web services, i.e. API—easy
  - run in a cluster behind load balancer
    - do backwards compatible releases
      - different versions of the same application will be running side-by-side
- Persistence services, like PostgreSQL—not as easy
  - if possible at all—out of scope of this talk

Rolling update of cluster nodes

- Nodes replaced one-by-one or in batches (Node 3)
- Can specify minimum number of node instances in service (Node 1)
- Can use connection draining to wait for in-flight requests (Node 2)



Docker on AWS

- Amazon EC2 Container Service (ECS)

https://aws.amazon.com/ecs/

- Docker for AWS

Docker for AWS is installed with a CloudFormation template that configures Docker in swarm-mode, running on instances backed custom AMIs

https://docs.docker.com/docker-for-aws/

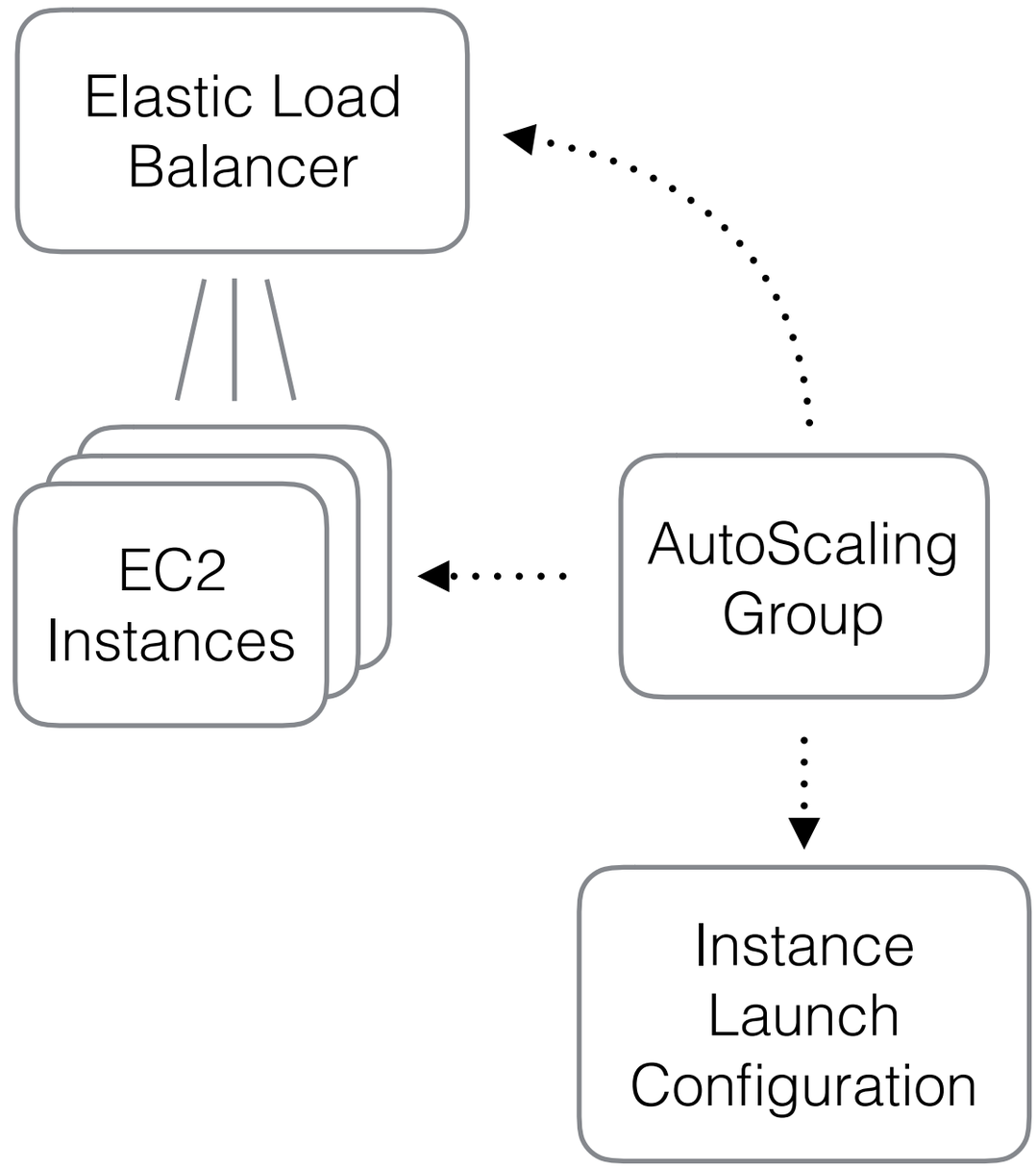
- Do It Yourself — 👍

DIY Options

- Build infrastructure
  - Using AWS Console (Web UI, manually)
  - AWS API / CLI + Bash + AWS CloudFormation
  - Docker Machine
  - Infrastructure as code (Terraform, Kubernetes, etc.)
- Setup deployment, configuration, etc. on top of it

Building a cluster with AWS

- AutoScaling Group (ASG)
  - uses instance launch configuration as a template for EC2 instances (AMI, instance type, etc.)
  - creates EC2 instances and registers them with ELB
  - maintains desired number of instances in a healthy state



Use AWS CodeDeploy for Deployment

- You do not need to make changes to your existing code to use it
- Deploys applications from Amazon S3 buckets and GitHub repositories
- Integrated with other AWS services, i.e. AutoScaling Groups

CodeDeploy Revision

- Revision is a folder with `appspec.yml`

```yaml
version: 0.0
os: linux
#files:
#permissions:
hooks:
  ApplicationStop:
    - location: elb/deregister_from_elb.sh
      timeout: 3600
    - location: stop.bash
      timeout: 3600
#  BeforeInstall:
  AfterInstall:
    - location: pull.bash
      timeout: 3600
    - location: decrypt-properties.bash
      timeout: 3600
  ApplicationStart:
    - location: up.bash
      timeout: 3600
    - location: elb/register_with_elb.sh
      timeout: 3600
  ValidateService:
    - location: validate.bash
      timeout: 3600
```

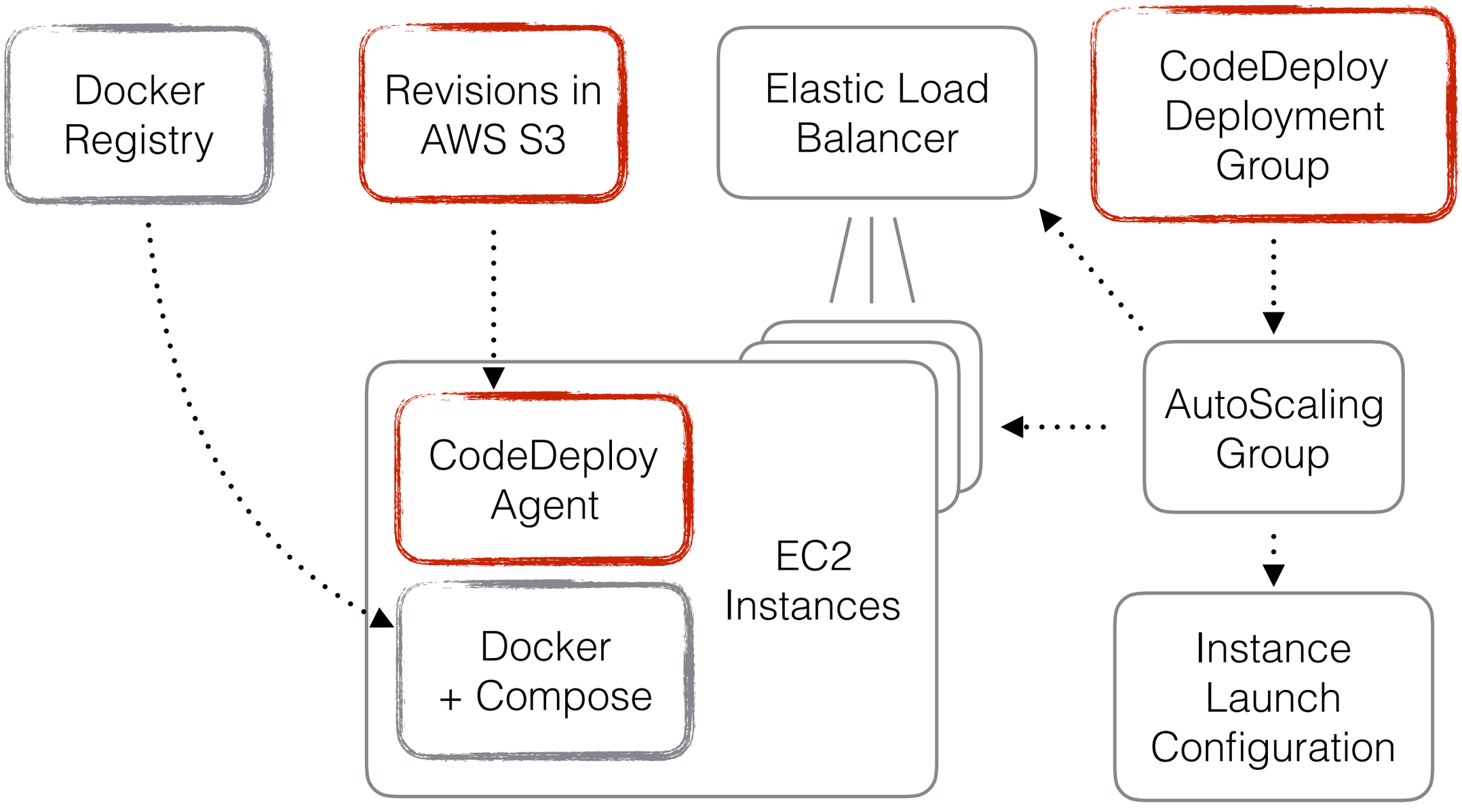
```
$ tree devops/build/revisions/my-app-v1
devops/build/revisions/my-app-v1
├── appspec.yml
├── compose.bash
├── compose.env
├── data
│   ├── logback.appenders.xml
│   ├── logback.xml
│   └── my-app-v1.properties
├── decrypt
├── decrypt-file.py
├── decrypt-properties.bash
├── docker-compose.my-app-v1.yml
├── elb
│   ├── README.md
│   ├── appspec.yml
│   ├── common_functions.sh
│   ├── deregister_from_elb.sh
│   ├── register_with_elb.sh
│   ├── start_httpd.sh
│   └── stop_httpd.sh
├── pull.bash
├── stop.bash
├── up.bash
└── validate.bash

2 directories, 21 files
```

`appspec.yml` + `docker-compose` = 💕

- Build a docker image for your app
- Publish it to registry to make it accessible from EC2
- Put your `docker-compose.yml` files to a revision
- Run `docker-compose stop/up/etc.` from `appspec.yml`'s hooks

CodeDeploy in the cluster



CodeDeploy Configuration

- Specify minimum healthy instances that must be healthy during a deployment:
  - in host count or percent

- or use `CodeDeployDefault.OneAtATime`

- Supports automatic rollback to previous successful version if deployment fails

Demo

https://github.com/anjlab/techtalks/tree/master/2017-02-03-aws-docker

Q/A

Dmitry Gusev
Software Engineer, AnjLab
@dmitrygusev