AUTO1 GROUP

Deploying ECS autoscaling app with Jenkins and Terraform

Konstantin Shcherban

Senior DevOps Engineer at AUTO1 GROUP

8 years of experience

Infrastructure, monitoring, deployments, etc.

About

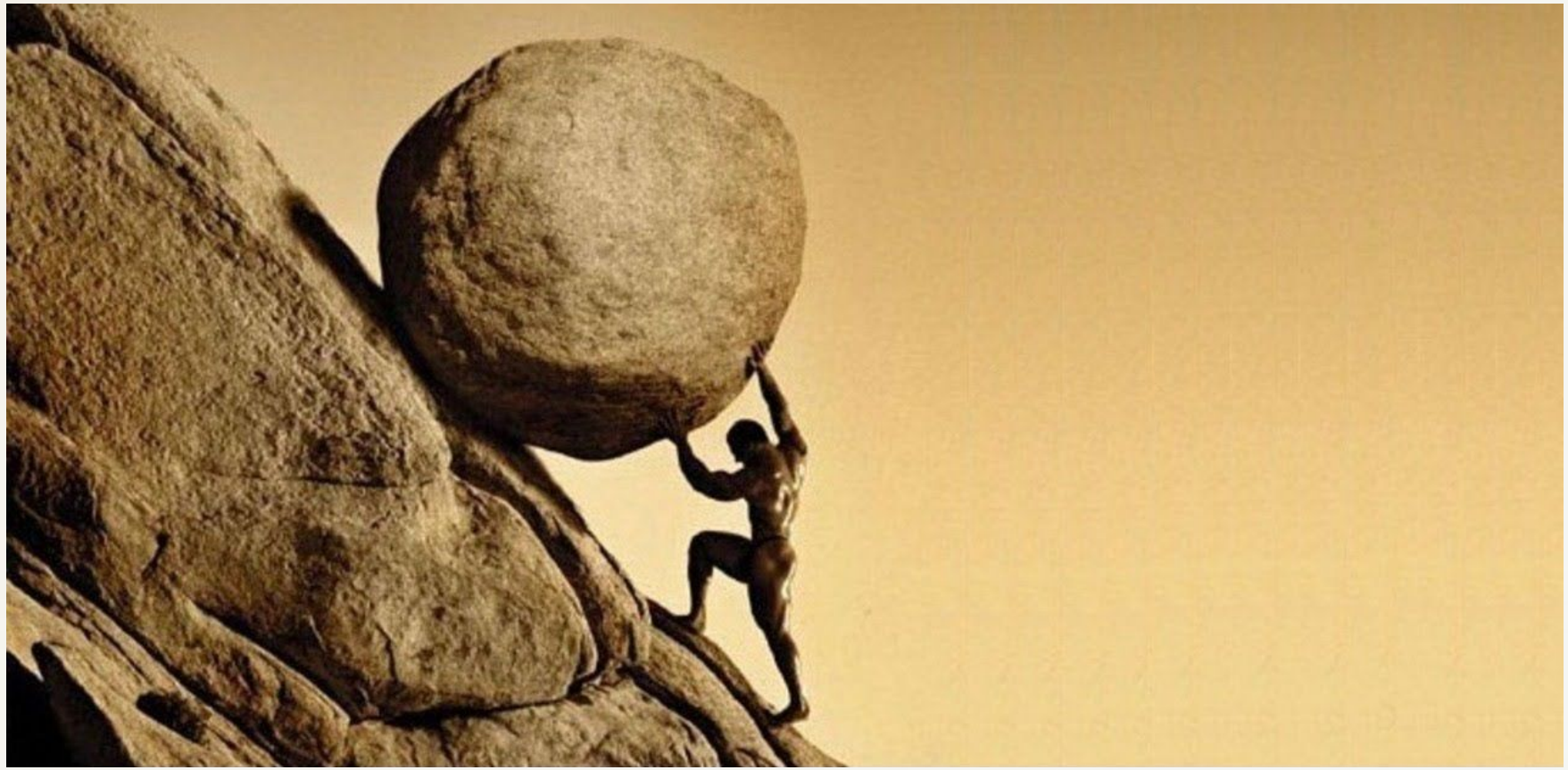
Headquartered in Berlin, AUTO1 Group is Europe's leading car trading platform.

- Diverse team of over 3,500 employees
- More than 55 nationalities
- Over 1,000,000 private customers
- Over 45,000 professional partners.

The brand independent automotive company owns business units like AUTO1.com, wirkaufendeinauto.de and Autohero.

The Co-Founders
Christian Berterr
and Hakan Koç

Managing service is tough



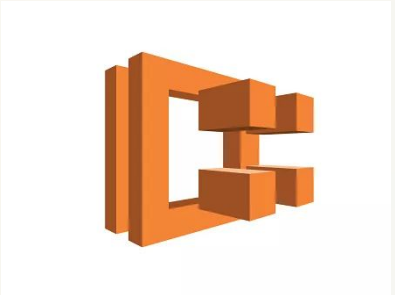
Managing microservices is tougher

AUTO1 | GROUP

What is AWS ECS?

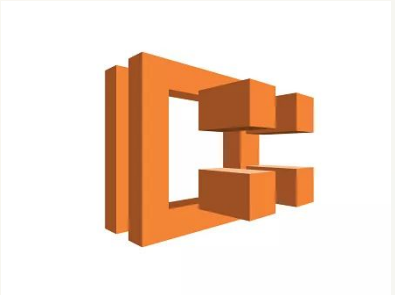
Why should you use it?

AWS ECS



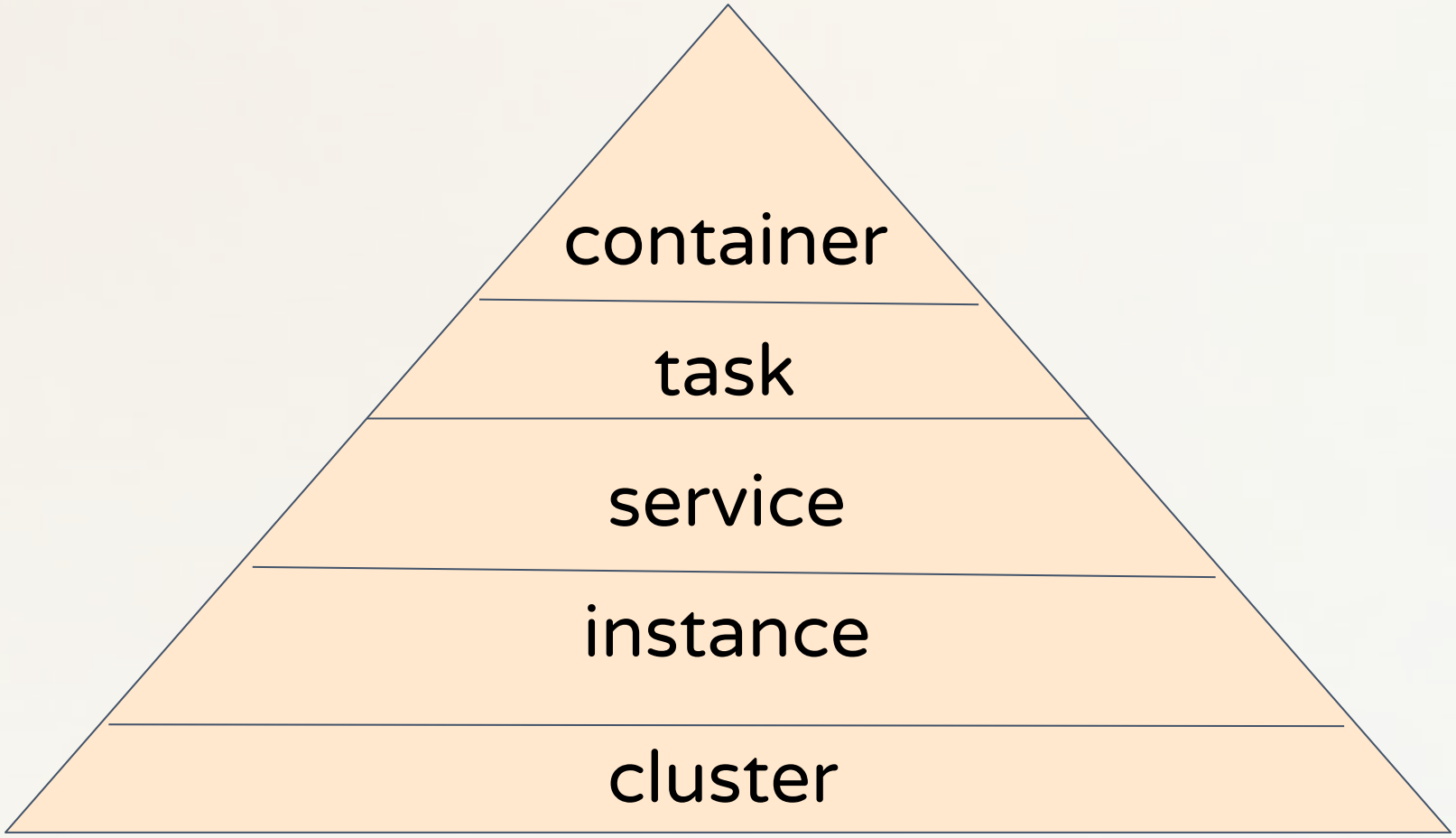
- black box container scheduler
- best integration with AWS services (even better than EKS)
- both managed infra *EC2* and serverless *Fargate*

AWS ECS

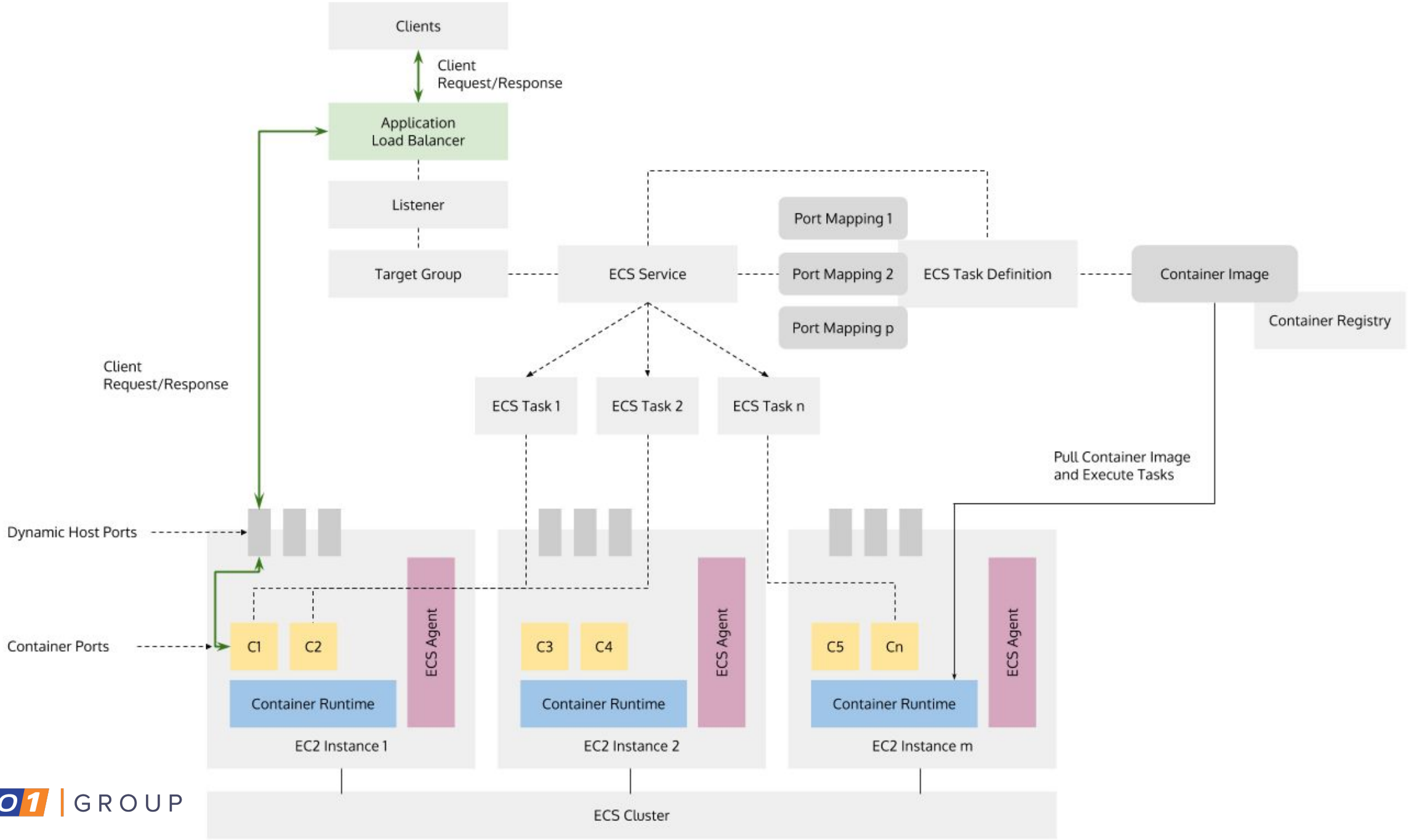


- pay only for instances you manage,
  no weird pricing like EKS, scheduler is free
- **stable**, AWS uses it internally
- not many features unlike vanilla Kubernetes

ECS architecture



AUTO1 | GROUP


Clients
Client
Request/Response
Application
Load Balancer
Listener
Target Group
ECS Service
Port Mapping 1
Port Mapping 2
Port Mapping p
ECS Task Definition
Container Image
Container Registry
Client
Request/Response
ECS Task 1
ECS Task 2
ECS Task n
Pull Container Image
and Execute Tasks
Dynamic Host Ports
Container Ports
C1
C2
C3
C4
C5
Cn
ECS Agent
ECS Agent
ECS Agent
Container Runtime
Container Runtime
Container Runtime
EC2 Instance 1
EC2 Instance 2
EC2 Instance m
ECS Cluster

ECS services

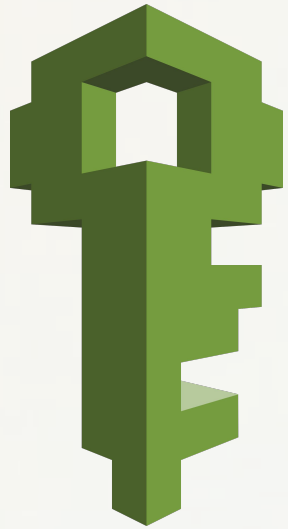
- Daemon
- Replica
- Batch task

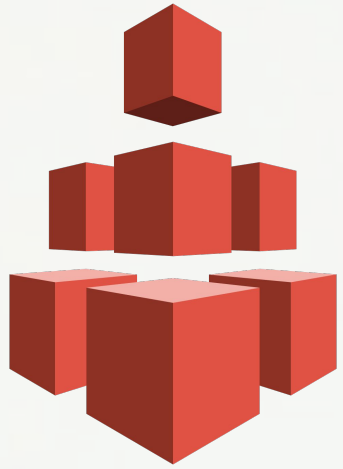
Do you really know AWS?

https://goo.gl/qVkWNp

AUTO1 | GROUP

Why to use ECS?



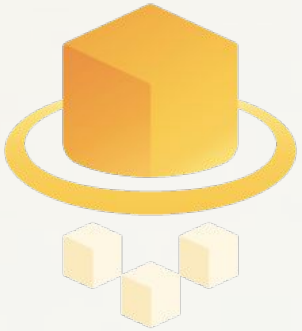


- ALB/NLB balancers
- route53 service discovery
- IAM task roles
- EBS/EFS volumes

AUTO1 | GROUP

Why to use ECS?

- Cloudwatch metrics/logs
- VPC networking, security groups per service
- ECS autoscaling
- ECR
- Serverless



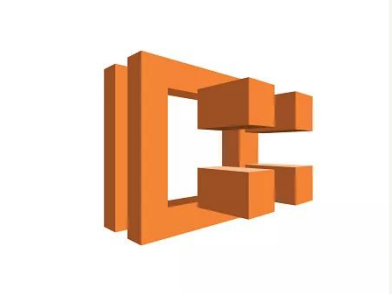
Why *NOT* to use ECS?

- full control over cluster wanted
- features needed that ECS doesn’t support
- fatal flaw — you didn’t write it
  (in case you have own container scheduler)

Autoscaling

- EC2 autoscaling

- ECS autoscaling



… they are **not** friends

Autoscaling

https://aws.amazon.com/blogs/compute/how-to-automate-container-instance-draining-in-amazon-ecs/

AWS Lambda

AUTO1 | GROUP

Deployment, Sir

+

HashiCorp
Terraform

What to do before deployment?

- think about service discovery
- configure logs and metrics collection
- determine where to store secrets

AUTO1 | GROUP

Service Discovery

- native ECS route53 SRV
- ALB/NLB as entrypoint
- Consul
- custom solution

AUTO1 | GROUP

Logs

- Cloudwatch $$$
- Elasticsearch, Logstash, Kibana
- third-party SaaS
- custom solution

AUTO1 | GROUP

Secrets management

- Hashicorp Vault
- S3 bucket
- AWS Systems Manager Parameter Store
- AWS Secrets Manager
- environment variables in task definitions
- custom solution

Deployment flow

1. Jenkins runs CI tests
2. Jenkins creates docker image and pushes it to ECR
3. Terraform deploys image to QA environment
4. QA and Devs test new QA deployment
5. Terraform deploys the same image to PROD

Docker image tagging

- tag latest image with latest tag
- tag all images with short git commit hash (`c656597`)
- if commit contains git tag, apply the tag (`v1.0.1`)
- tag all images deployed to PROD with prefix (`prod_`)

ECR image policies

- apply IAM policies to restrict image access
- apply lifecycle policies
  (AWS limits 1000 images per repo)
- apply sharing policies with other accounts if needed

ECR image policies

```
"rules": [
        {
            "rulePriority": 1,
            "description": "Keep prod tags",
            "selection": {
                "tagStatus": "tagged",
                "tagPrefixList": ["prod", "v", "latest"],
                "countType": "imageCountMoreThan",
                "countNumber": 100
            },
            "action": {
                "type": "expire"
            }
        },
```

AUTO1 | GROUP

Challenges

- Unification of services, especially between different teams
- Logs/metrics collection per service/container
- Security
- Alerting and resource tagging

Challenges

- LB has limit of 100 rules per listener
- update of large EC2 autoscaling group
- make cluster resource reservation == utilization

Demo time

github.com/auto1-oss/aws-ecs-jenkins-terraform

AUTO1 | GROUP

Thank you. Questions?

Konstantin Shcherban

Email: insider@prolinux.org

Github: github.com/kshcherban

Blog: www.prolinux.org (mostly in Russian)