Инфраструктурные платформы

Не забудь включить запись!

План

- Пара слов о курсе
- Что такое инфраструктурная платформа?
- Как она помогает компаниям?
- Почему Kubernetes – хорошая основа для платформы?

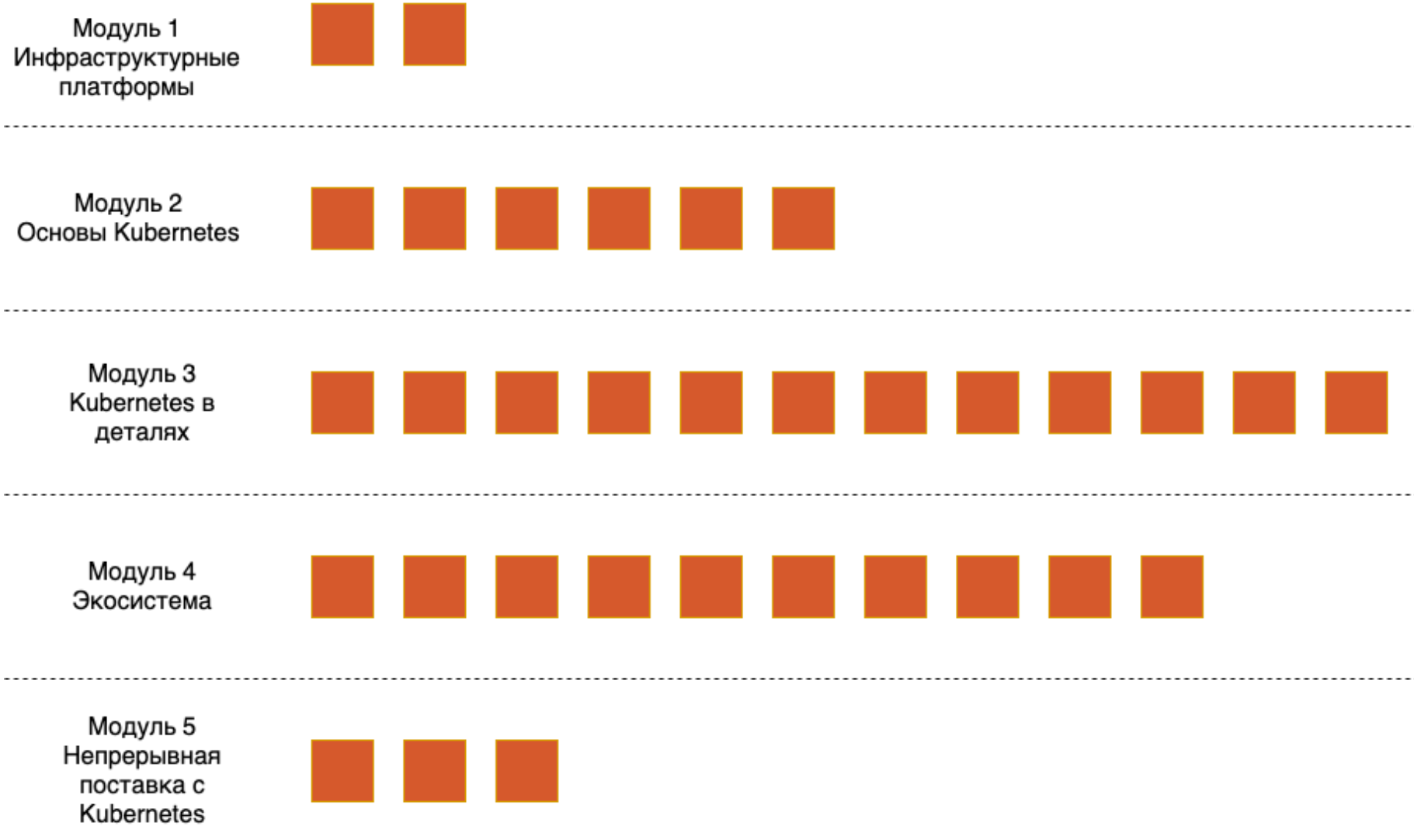
О курсе

- Авторы и преподаватели курса
- Чем курс будет вам полезен
- Состав курса и формат

Экспресс 42

Среди наших клиентов

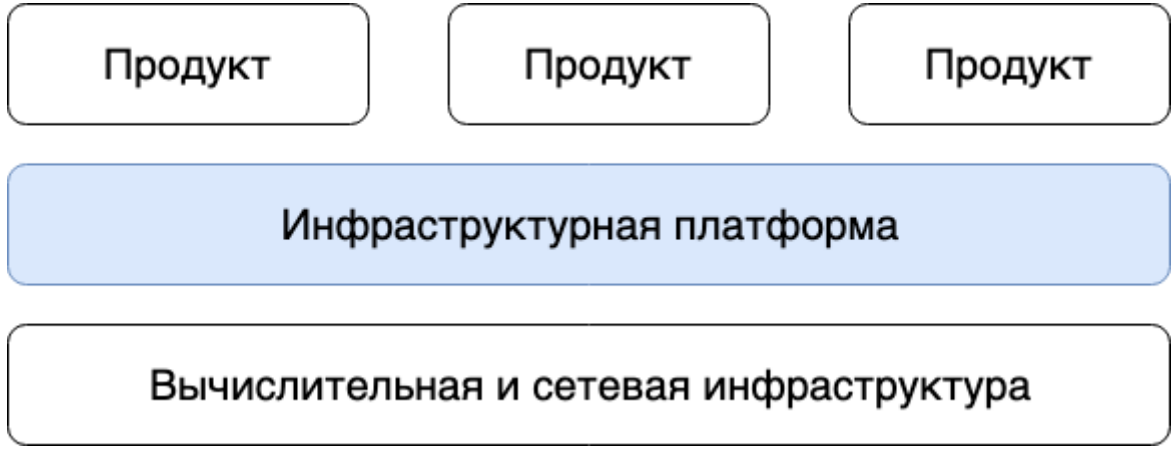
Из курса вы узнаете



Инфраструктурные платформы

Определение

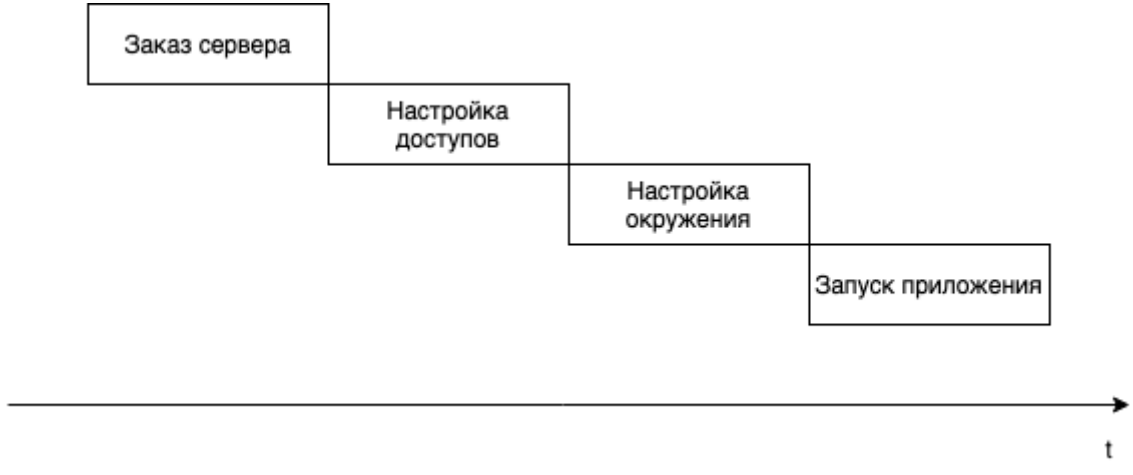
Инфраструктурная платформа – совокупность систем, сервисов, людей, практик и соглашений, нацеленная на обеспечение этапов разработки и эксплуатации цифровых продуктов.



Традиционный подход

Пример: продуктовой команде понадобилось новое окружение

Каждый этап выполняется разными сотрудникамим/командами



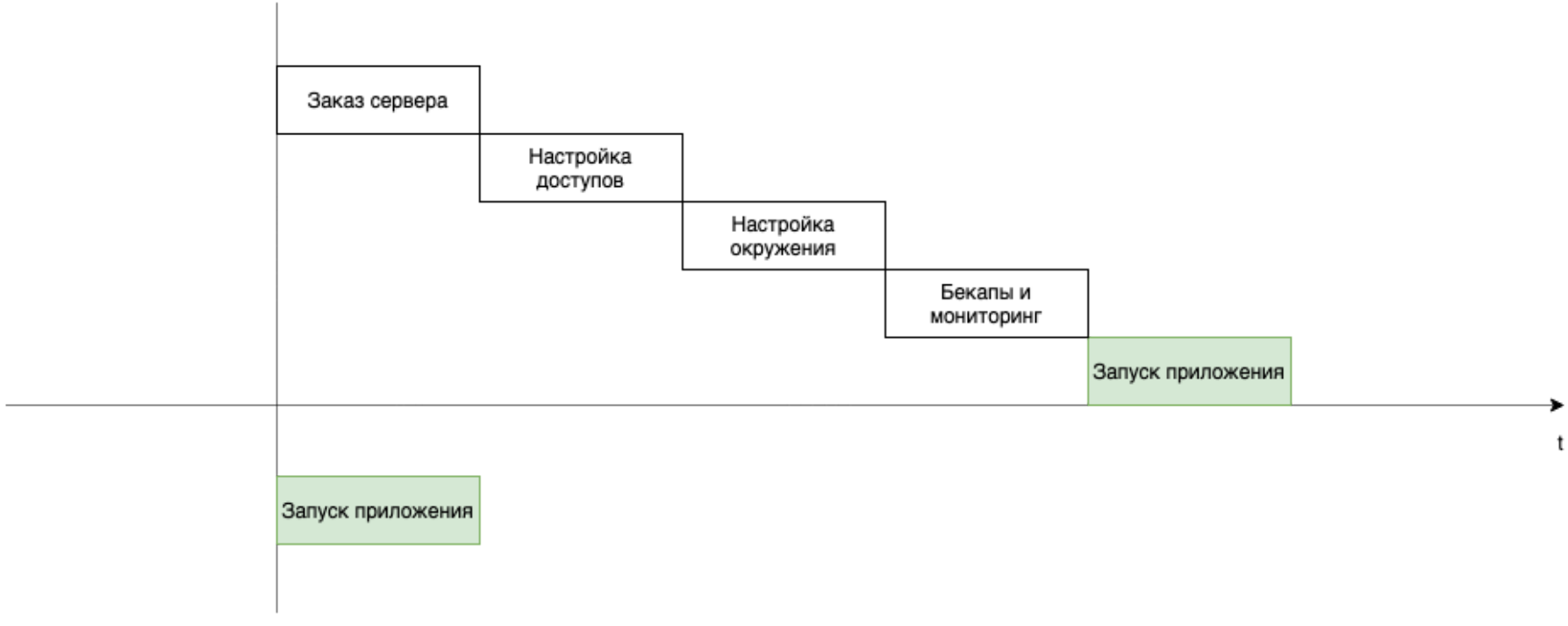
Традиционный подход – орг. структура

Эти сотрудники не заинтересованы в благополучии продукта

У них своя очередь задач и пожаров и свои KPI

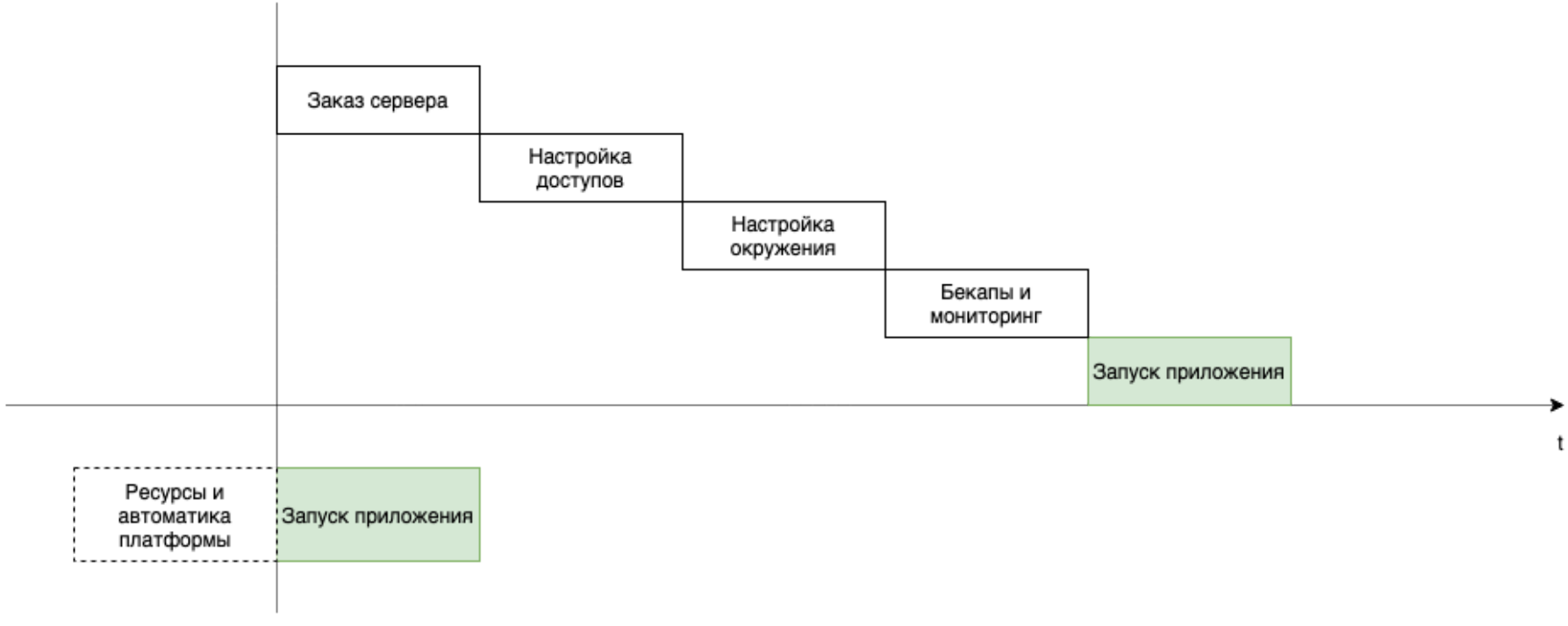
Традиционный подход – скорость

Вызов API быстрее тикетов в таск-треккере

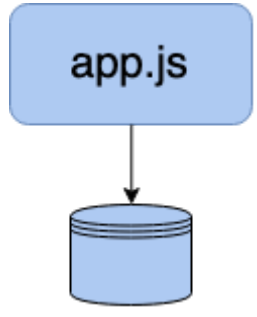


Традиционный vs Платформенный

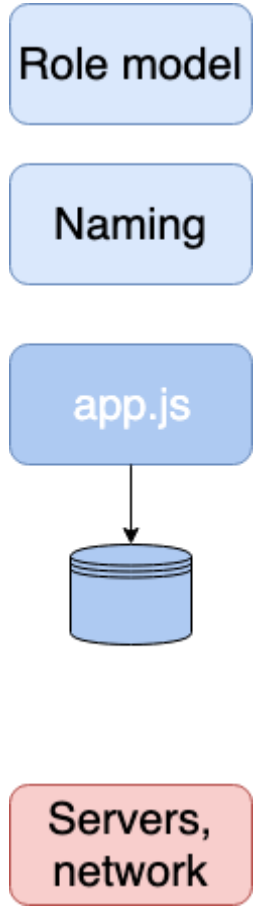
Процедуры разработаны и автоматизированы заранее



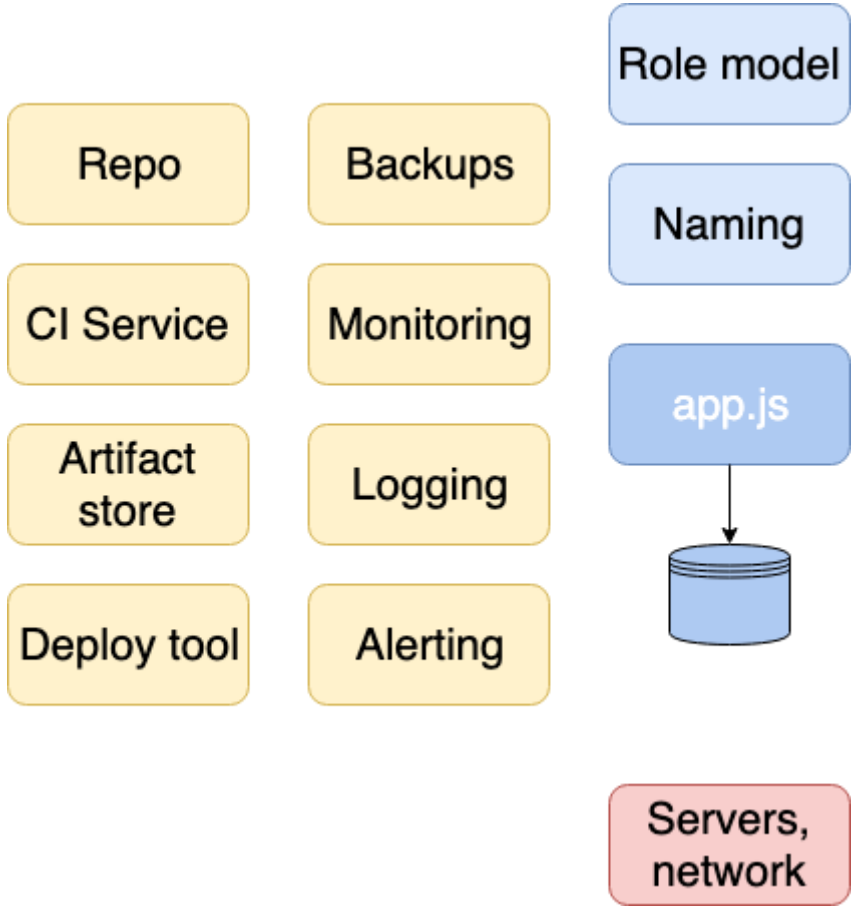
Традиционный подход – экосистема



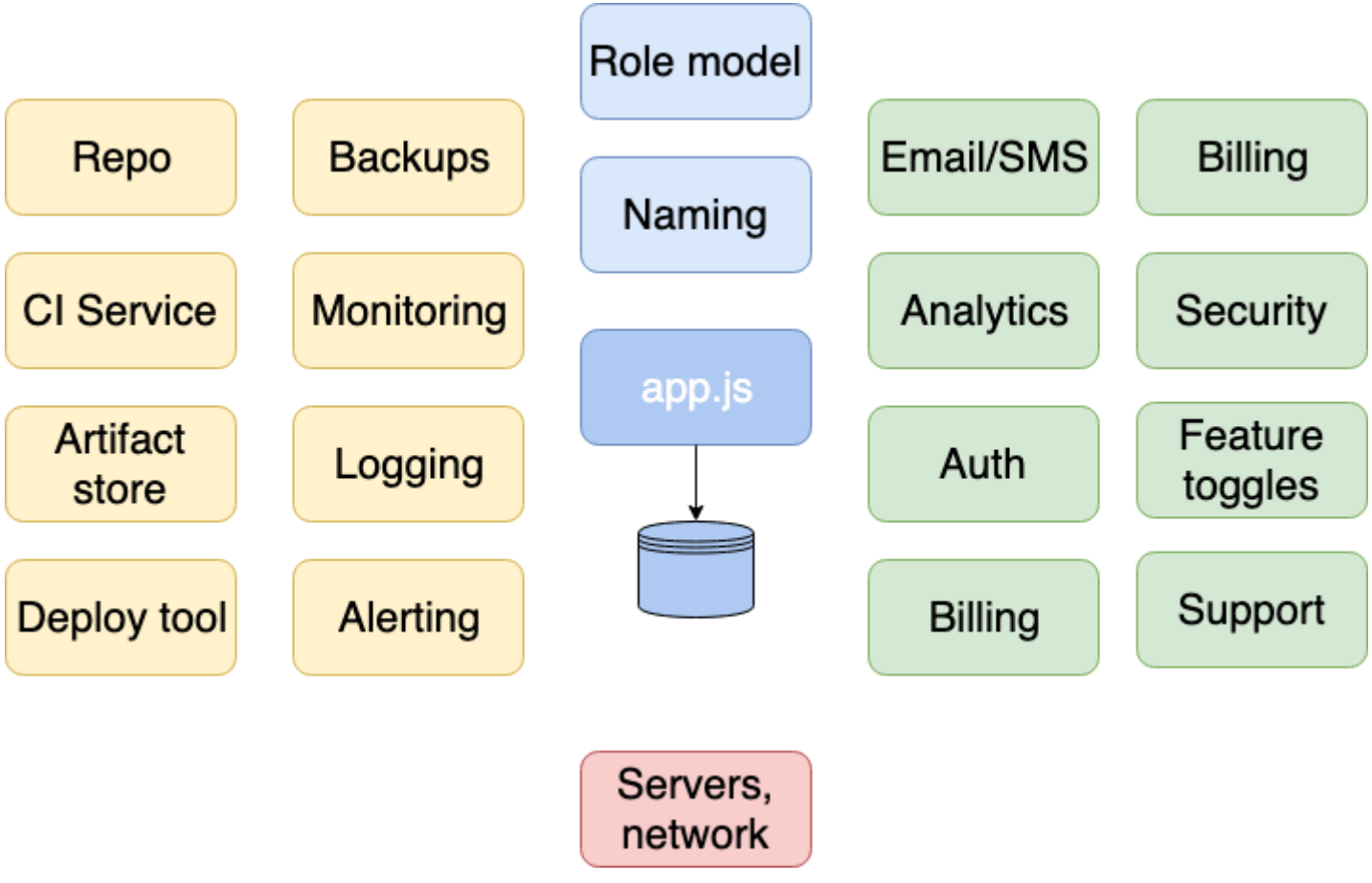
Традиционный подход – экосистема



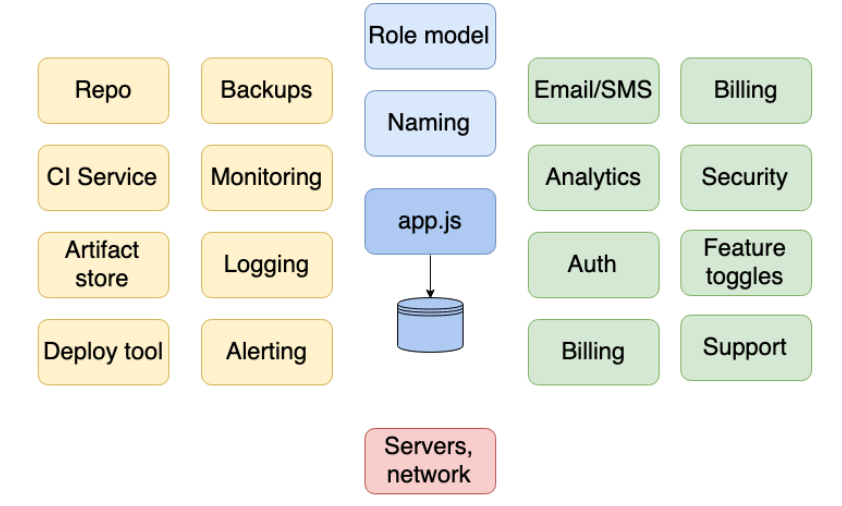
Традиционный подход – экосистема

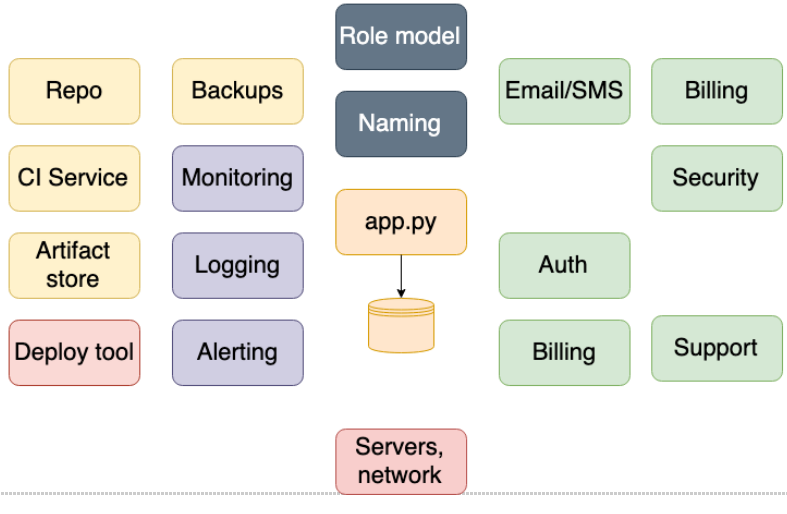


Традиционный подход – экосистема



Такое повторяется в каждой команде





Чем это плохо

- Продуктовая команда отвлекается от своего продукта
- Требуются дополнительные компетенции
- Дополнительная когнитивная нагрузка
- Пересечения зон ответственности
- Сложее быстро принимать решения
- Большое количество необходимых коммуникаций
- Затруднено распространение хороших практик и опыта
- Затруднено внедрение стандартов

Что на этот счет думает индустрия?

- Опыт Экспресс 42
- Публикации и доклады
- Исследования
- Провайдеры

DORA State of DevOps

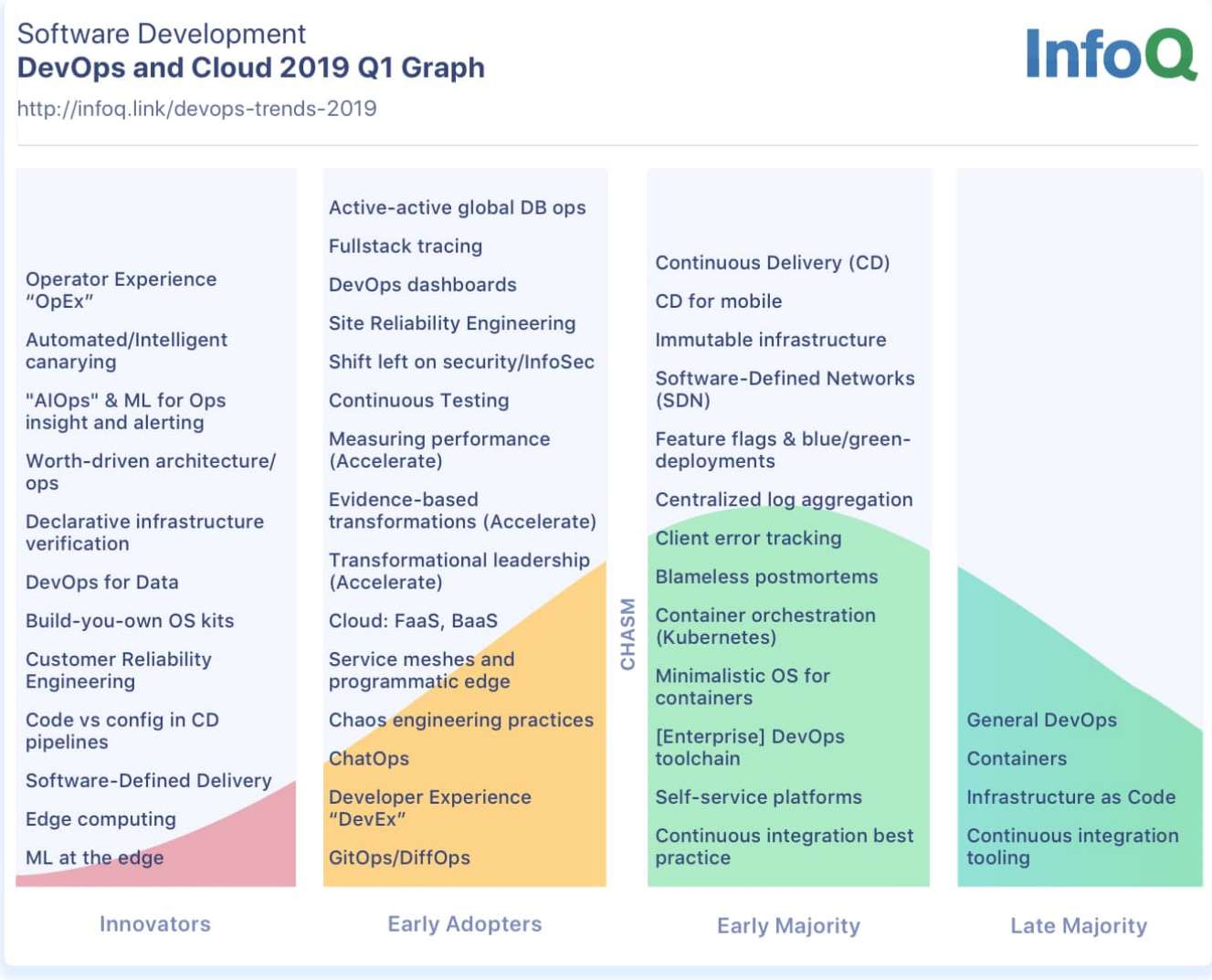
Platform as a Service

Another way to provide a better service to application developers is through implementing a platform as a service (PaaS) in which "the consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment."[16] Examples of a PaaS include Heroku, RedHat OpenShift, Azure App Service, Google App Engine, AWS Elastic Beanstalk and Cloud Foundry.

Only 24 percent of respondents report using a PaaS. However, respondents that do most of their work on a PaaS are 1.5 times more likely to be in the elite performance group. These respondents agreed or strongly agreed that their team uses libraries and infrastructure defined by the PaaS as the basis for their applications, can deploy their application into the cloud on demand using a single step, and can perform self-service changes on-demand for databases and other services required by their application.

[16] NIST Special Publication 800-145: "The NIST Definition of Cloud Computing."

InfoQ DevOps Trends



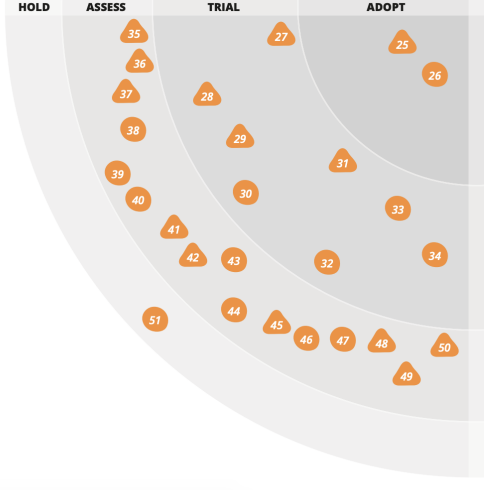
ThoughtWorks Technology radar

PLATFORMS

ADOPT
25. .NET Core
26. Kubernetes

TRIAL
27. Azure
28. Contentful *NEW*
29. EMQ *NEW*
30. Flood IO
31. GKE
32. Google Cloud Platform
33. Keycloak
34. WeChat

ASSESS
35. AWS Fargate *NEW*
36. Azure Service Fabric
37. Azure Stack *NEW*
38. Cloud Spanner
39. Corda
40. Cosmos DB
41. Godot *NEW*
42. Interledger *NEW*
43. Language Server Protocol
44. LoRaWAN
45. Mongoose OS *NEW*
46. Netlify
47. TensorFlow Serving
48. TICK Stack *NEW*
49. Web Bluetooth *NEW*
50. Windows Containers



Gartner How to Scale DevOps by Building Platform Teams

Gartner

Licensed for Distribution

How to Scale DevOps by Building Platform Teams

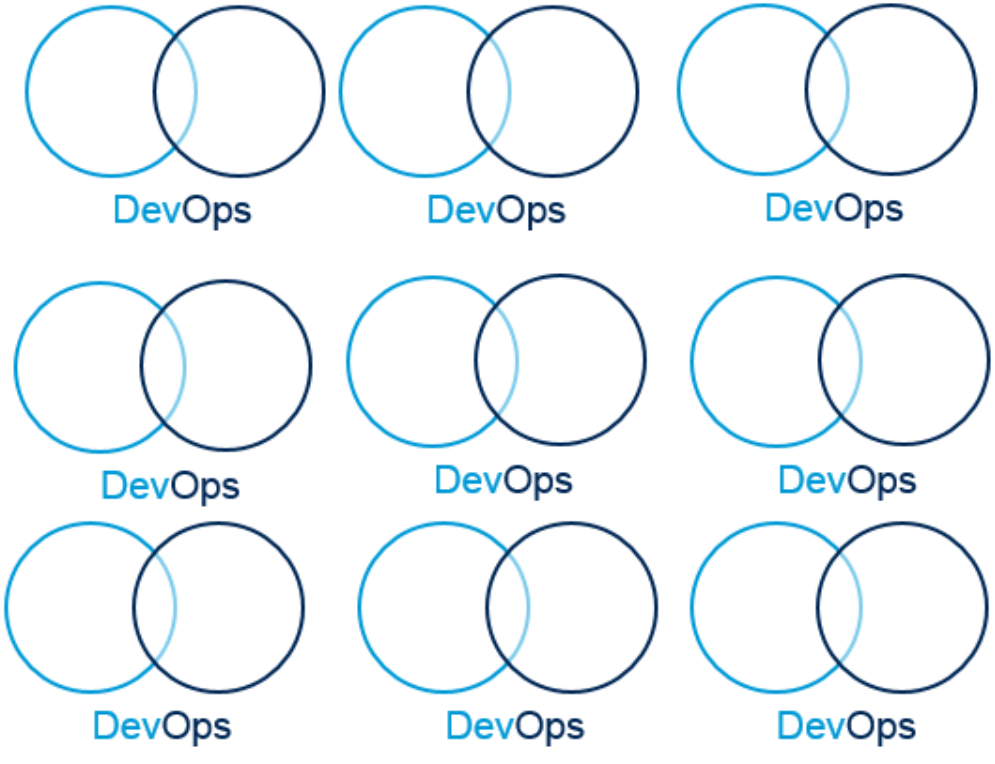
Published 8 April 2019 - ID G00382537 - 13 min read

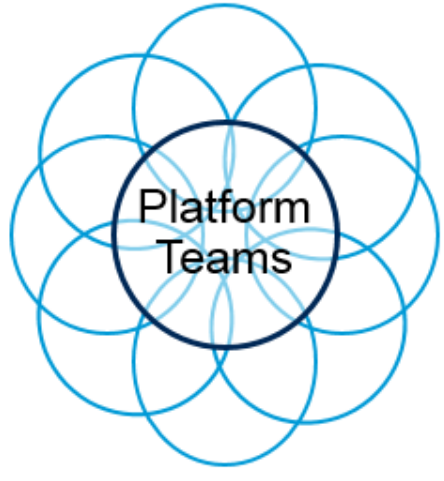
By Analysts Daniel Betts, George Spafford

Enterprises that achieve initial success with DevOps initiatives often struggle to scale their efforts. I&O leaders should dedicate teams to operating shared self-service platforms that enable agile product teams to accelerate delivery while ensuring quality and standardization.

Gartner How to Scale DevOps by Building Platform Teams

DevOps Team Topologies Create Problems as They Scale Up

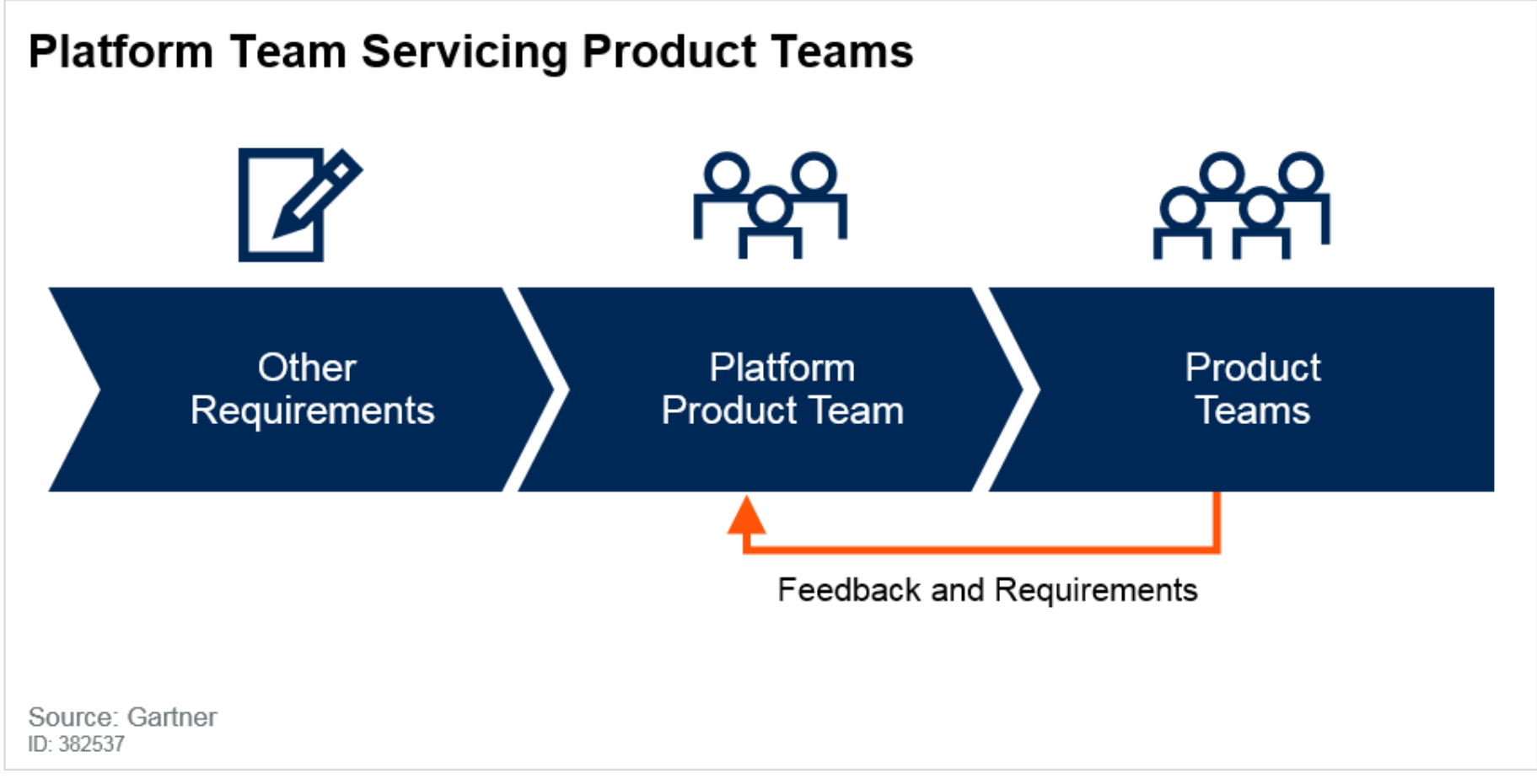




Product Teams

Source: Gartner
ID: 382537

Gartner How to Scale DevOps by Building Platform Teams

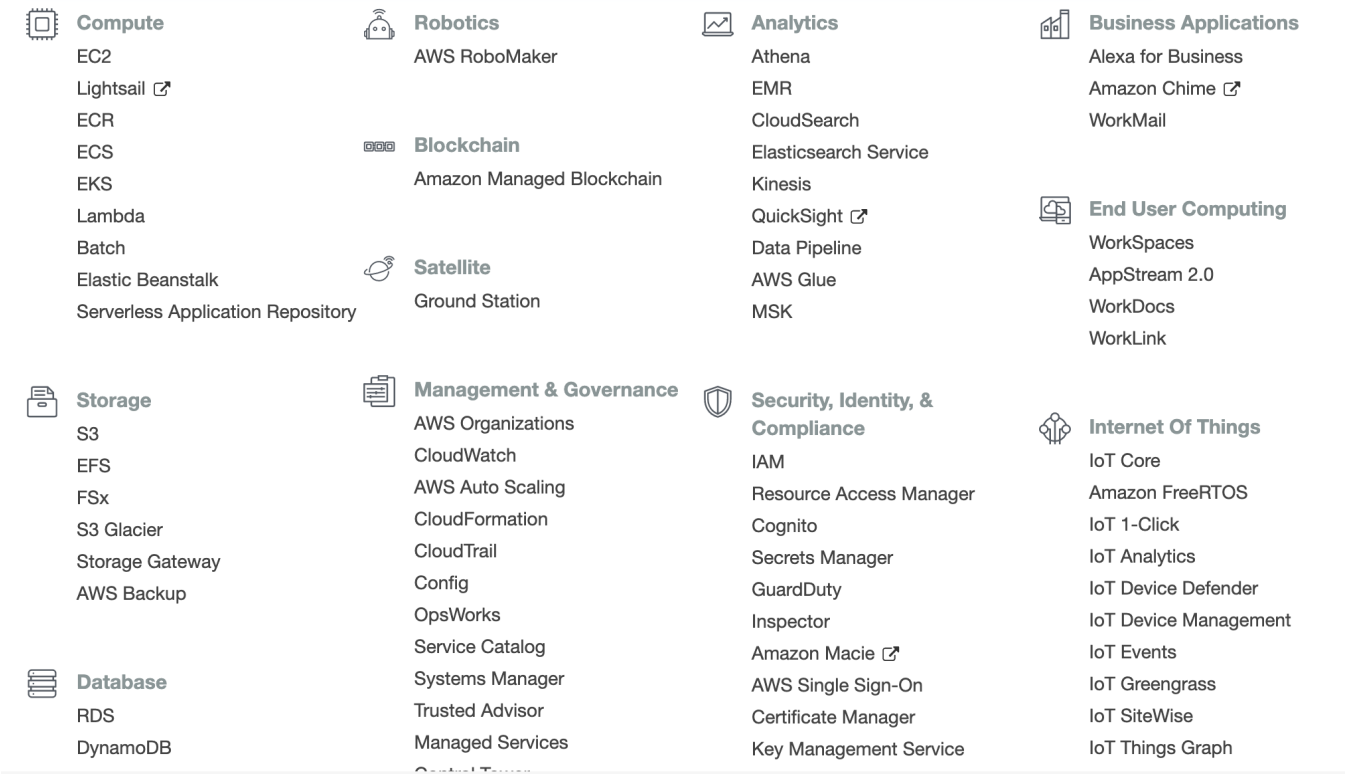


Gartner How to Scale DevOps by Building Platform Teams

> By 2023, 90% of enterprises will fail to scale DevOps initiatives if shared self-service platform approaches are not adopted.

Облачные провайдеры

- Heroku
- AWS
- GCP
- Azure
- Yandex
- Mail.ru



Compute
EC2
Lightsail
ECR
ECS
EKS
Lambda
Batch
Elastic Beanstalk
Serverless Application Repository

Storage
S3
EFS
FSx
S3 Glacier
Storage Gateway
AWS Backup

Database
RDS
DynamoDB

Robotics
AWS RoboMaker

Blockchain
Amazon Managed Blockchain

Satellite
Ground Station

Management & Governance
AWS Organizations
CloudWatch
AWS Auto Scaling
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Systems Manager
Trusted Advisor
Managed Services

Analytics
Athena
EMR
CloudSearch
Elasticsearch Service
Kinesis
QuickSight
Data Pipeline
AWS Glue
MSK

Security, Identity, & Compliance
IAM
Resource Access Manager
Cognito
Secrets Manager
GuardDuty
Inspector
Amazon Macie
AWS Single Sign-On
Certificate Manager
Key Management Service

Business Applications
Alexa for Business
Amazon Chime
WorkMail

End User Computing
WorkSpaces
AppStream 2.0
WorkDocs
WorkLink

Internet Of Things
IoT Core
Amazon FreeRTOS
IoT 1-Click
IoT Analytics
IoT Device Defender
IoT Device Management
IoT Events
IoT Greengrass
IoT SiteWise
IoT Things Graph

История Kubernetes

Посмотрим на Google

- Огромное количество продуктов
- Высокие требования к доступности и качеству
- Инфраструктура по всему миру
- Прочие сложности, которые проявляются на масштабе

Google Borg

Large-scale cluster management at Google with Borg

Abhishek Verma† Luis Pedrosa‡ Madhukar Korupolu
David Oppenheimer Eric Tune John Wilkes

Google Inc.

Abstract

Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines.

It achieves high utilization by combining admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. It supports high-availability applications with runtime features that minimize fault-recovery time, and scheduling policies that reduce the probability of correlated failures. Borg simplifies life for its users by offering a declarative job specification language, name service integration, real-time job monitoring, and tools to analyze and simulate system behavior.

We present a summary of the Borg system architecture and features, important design decisions, a quantitative analysis of some of its policy decisions, and a qualitative examination of lessons learned from a decade of operational experience with it.

1. Introduction

The cluster management system we internally call Borg admits, schedules, starts, restarts, and monitors the full range of applications that Google runs. This paper explains how.

Borg provides three main benefits: it (1) hides the details of resource management and failure handling so its users can focus on application development instead; (2) operates with very high reliability and availability, and supports applications that do the same; and (3) lets us run workloads across tens of thousands of machines effectively. Borg is not the first system to address these issues, but it's one of the few operating at this scale, with this degree of resiliency and com-

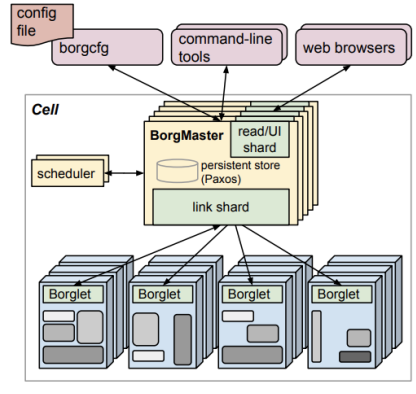


Figure 1: The high-level architecture of Borg. *Only a tiny fraction of the thousands of worker nodes are shown.*

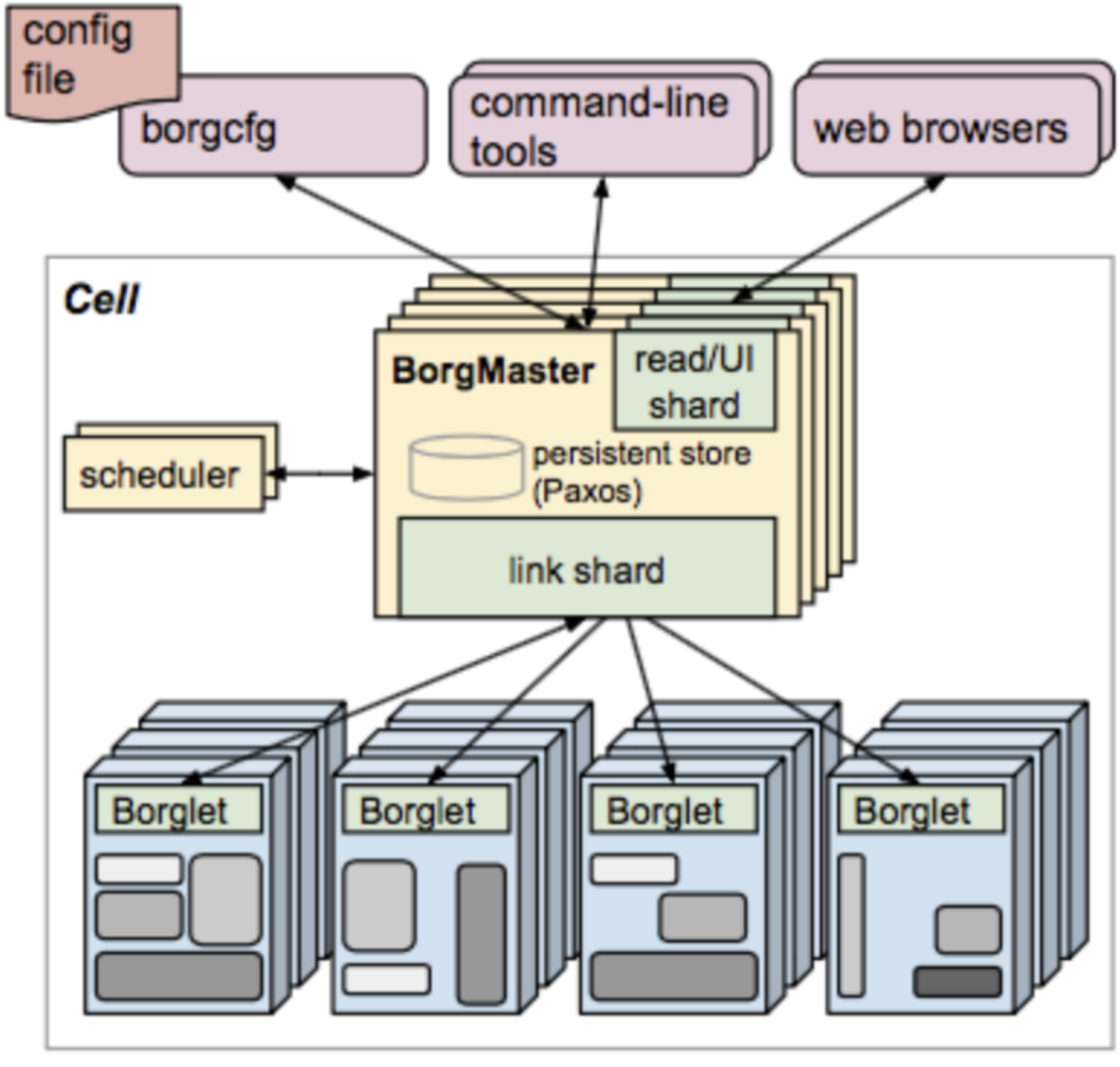
cluding with a set of qualitative observations we have made from operating Borg in production for more than a decade.

2. The user perspective

Borg's users are Google developers and system administrators (site reliability engineers or SREs) that run Google's applications and services. Users submit their work to Borg in the form of *jobs*, each of which consists of one or more *tasks* that all run the same program (binary). Each job runs in one Borg *cell*, a set of machines that are managed as a unit. The remainder of this section describes the main features exposed in the user view of Borg.

2.1 The workload

Google Borg


config file
borgcfg
command-line tools
web browsers
Cell
BorgMaster
read/UI shard
persistent store (Paxos)
scheduler
link shard
Borglet
Borglet
Borglet
Borglet

Google Borg

Декларативное описание для приложений

```
job hello_world = {
    runtime = { cell = 'ic' }
    binary = '../application-binary'
    args = { port = '%port%' }
    requirements = {
        RAM = 4G
        disk = 50G
        CPU = 2
    }
    replicas = 5
}
```

Kubernetes

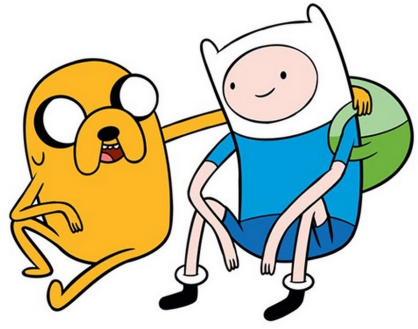
- k8s
- Переосмысление проекта **Borg** от Google в контексте контейнеров
- **~55 000** звезд на GitHub
- **~80 000** коммитов
- Более 3 лет в Open Source
- Версия 1.0 была выпущена 21 июля 2015
- Тогда же Kubernetes присоединился к CNCF и стал первым выпускником

Kubernetes – основа для платформы

- Новый подход к управлению инфраструктурой
- Унификация описания инфраструктуры
- Современные подходы к мониторингу и логированию
- Реализация прочих DevOps-практик
- Множество встроенных механизмов
- Огромная экосистема и сообщество

Подводя итоги

- Инфраструктурная платформа помогает продуктовым командам
- Инфраструктурная платформа – ваш внутренний продукт
- Платформенная команда формирует практики и стандарты
- Со временем инфраструктурная платформа становится вашим конкуретным преимуществом
- Kubernetes – достойная основа для платформы
- Традиционные подходы к разработке и эксплуатации придется переосмыслить



Спасибо за внимание!

Время для ваших вопросов!