What is a Platform?

A platform encompasses a span of technologies touched by different engineering teams.

Platform can refer to physical or virtual infrastructure and networking.

A company’s platform also includes telemetry, application delivery, container deployment and orchestration with a tool such as Kubernetes, content delivery networks (CDNs) and CI/CD tooling.

In the broadest sense, the platform is the environment and set of technologies upon which a company builds, deploys and delivers its applications.
Platform as a Product

Explicit API Boundary

Platform Managed by Platform Team

Backend Infrastructure
What a Platform Team Does?

Platform teams craft and curate a portfolio of technologies to maximize the efficiency of their engineering and networking teams, and establish best practices so the organization can scale more easily and securely.

Platform Ops teams seek not to limit choices, but to drive consensus and ensure that everyone gets the tools and capabilities they need.

Platform Ops walks the fine line of crafting and evangelizing a well-honed menu of choices that can serve 95% of needs while remaining open to feedback on the choices.
Dal DevOps to PlatformOps

Platform Team by Team Topologies

QRC Team Topologies

Based on Team Topologies, QRC by Henny Portman, May 2020

Conway’s law: “Organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations.”

Team first approach: start with the team for effective software delivery. There are multiple aspects to consider and nurture: team size, team lifespan, team relationships, and team cognition.

Organizational sensing: expect to adapt and evolve your organization structure.

Scaling teams: Organizational groupings should follow Dunbar’s number, beginning with around 5-8 people, then increasing to around 15 people, then 50, then 150, then 500, and so on.

Brook’s law: “Adding new people to a team doesn’t immediately increase its capacity.”

Cognitive load: “The total amount of mental effort being used in the working memory.” Restrict team responsibilities to match the maximum team cognitive load.

• Intrinsic cognitive load – relates to aspects of the task fundamental to the problem space
• Extrinsic cognitive load – relates to the environment in which the task is being done
• Germane cognitive load – relates to aspects of the task that need special attention for learning or high performance

Interaction mode: Facilitating

Interaction mode: Collaborating

Interaction mode: X-as-a-Service

Interaction mode: X-as-a-Service

Enabling Team

Evolution of team topologies

Primary interaction modes for the 4 fundamental team topologies:

• Collaboration: working closely together with another team
• X-as-a-Service: consuming or providing something with minimal collaboration
• Facilitating: helping (or being helped by) another team to clear impediments

Stream-Aligned Team

Complicated-Subsystem Team

Platform team

Complicated-Subsystem Team: a team with a special remit for a subsystem that is too complicated to be dealt with by a normal stream-aligned team or platform team. Optional and only used when really necessary.

Platform team: a team that works on the underlying platform supporting stream-aligned teams in delivery. The platform simplifies otherwise complex technology and reduces cognitive load for teams that use it.

Stream-Aligned Team: a team aligned to the main flow of business change, with cross-functional skills mix and the ability to deliver significant increments without waiting on another team.
The Platform Simplifies Complex Technology

<table>
<thead>
<tr>
<th>Database</th>
<th>Streaming &amp; Messaging</th>
<th>Application Delivery &amp; Image Build</th>
<th>Continuous Integration &amp; Delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scheduling &amp; Orchestration</th>
<th>Management &amp; Security</th>
<th>Network Protocols &amp; Protocols</th>
<th>Service Proxy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>API Delivery</th>
<th>Service Mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Platform</th>
<th>Observability and Analytics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Certified Kubernetes - Distributors</th>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Certified Kubernetes - Cluster</th>
<th>Logging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Certified Kubernetes - Installer</th>
<th>Dealing Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Container Registry</th>
<th>Security &amp; Compliance</th>
<th>Key Management</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Automation &amp; Configuration</th>
<th>Container Registry</th>
<th>Security &amp; Compliance</th>
<th>Key Management</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Automation &amp; Configuration</th>
<th>Container Registry</th>
<th>Security &amp; Compliance</th>
<th>Key Management</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
By 2025, 95% of enterprises will fail to scale DevOps initiatives if shared self-service platform approaches are not adopted.

[Gartner]
Why DevOps Success Requires Platform Teams

I&O leaders find it difficult to provide enough operations expertise in DevOps product teams as they scale, resulting in slower delivery cycles, software defects and frustration.

I&O leaders are unable to ensure high standards of governance and production efficiency when product teams recreate platforms’ capabilities inconsistently from team to team.

Adopt a scalable approach to DevOps by establishing dedicated platform teams to rapidly respond to product team needs.

[Gartner]
Dal DevOps al PlatformOps

**Monolithic Architecture**

- User Interface
- Business Layer
- Data Interface

**Microservices Architecture**

- Microservice UI
- Microservice
- Microservice
- Microservice
What are Kubernetes operators?
Composable operators
Custom Resource Definition as a data model
Dal DevOps al PlatformOps

Krateo PlatformOps use cases

- PlatformOps for Platform Team
- Developer Portal for internal community
- FinOps for C-level managers
PlatformOps for Platform Team

**Platform Team provides services to internal consumers:**

- Automation is required
- Self-service catalog which exposes internal services
- Services can be anything: infrastructure, software templates, machine learning models, etc
- Services can be anywhere: on premise, public cloud, hybrid cloud, multi cloud

**BUT!**

- Automation for each service is specialized for that service
- The Day-2 operation is still to automate
- What about legacy environments?

**Which means:**

- Diverging automation streams
- Multiple tools, multiple skill sets
- Increasing cost and risk for PlatformOps
Consuming internal services quickly, safely and independently as possible:

▸ Self-service catalog which exposes internal services must be user friendly and reliable

▸ Every consumable service must require the minimal set of information that the end user could know

▸ All the lifecycle of the service and the relative data must be centralized

BUT!

▸ The common onboarding experience is more and more complex

▸ Understanding ownership of services and resources is complicated

Which means:

▸ Slower software lifecycles, duplicated services

▸ **Increasing costs and risks** for the business
FinOps for C-level managers

**FinOps** is shorthand for “Cloud Financial Operations” or “Cloud Financial Management” or “Cloud Cost Management”.

It defines an opportunity for everyone to take ownership of their cloud usage and manage their cloud costs.

However, the variable spend model of cloud is multi-dimensional, multi-layered, multi-service provider.

The opportunity was elusive – until Krateo: now you can optimize Platform costs across multiple providers.
Why Krateo

Bring your own Kubernetes
Krateo can be installed on any Kubernetes certified distribution by creating a secure by default infrastructure stack – no lock-in

Universal
Create and deploy your resources on Kubernetes or any other on premise, public, hybrid and multi cloud platform - without adding complexity or new tools

Self Service
Use the templates available or create new ad hoc ones to develop applications, models, databases, websites, microservices and everything you need – standardization enables quicker cycles

Flexible
Create any logical and physical component required by internal consumers with a composable approach – which means reuse and cost savings
Kratoe follows the Red Hat vision:
Any workload, any footprint, any location
Thank you

linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHat