cloudowski

5 LEVELS OF PLATFORM MATURITY

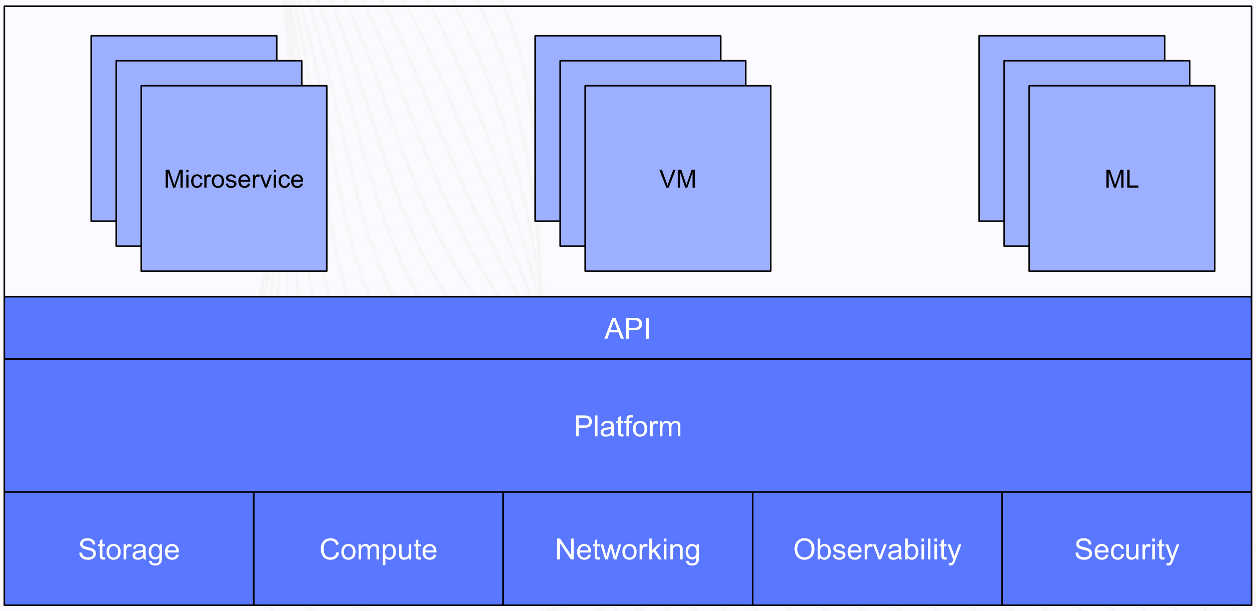
OVERVIEW

Prepared by
Tomasz Cholewa
www.cloudowski.com

Platform Maturity Levels

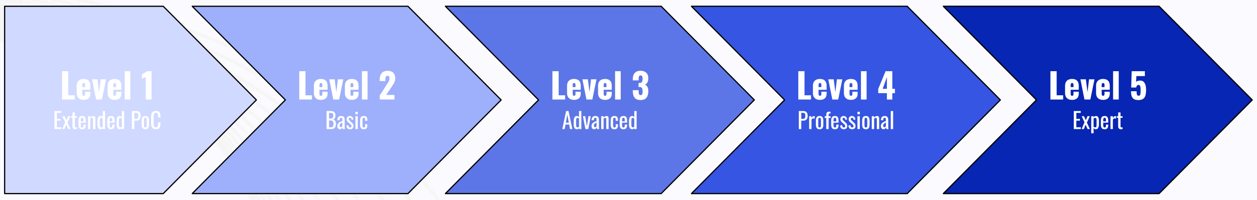
What is a platform?

A platform is a **complete environment** able to run all kinds of applications.
Modern platform these days is based on containers and **Kubernetes** (or **OpenShift**) running in the cloud, on on-premises hardware or in hybrid environments.
To provide necessary features, multiple **Cloud Native** technologies are used to accomplish high reliability, security and scalability.
Together with Platform Engineering and **DevOps** practices, the platform built within the organization makes it possible to develop and manage almost any software in a cost-effective manner.



What are Platform Maturity Levels™?

The **five levels** are distinguished stages of implementation of a specific subset of features. These features increase capabilities of the platform in an **incremental** manner. Each level includes strategies for achieving a specific level of maturity to provide more confidence in the platform and allows for running more critical software.



Why is this level-based approach better?

This approach is more **evolutionary** rather than revolutionary. It’s more **practical** and takes into account the time needed for learning new tools and processes.
It also allows the platform's capabilities to be better aligned with the **organization's context** (e.g. existing software, policies, restrictions, etc.).
Each level can introduce or extend the use of particular practice or technology (e.g. GitOps, Zero Trust Environment, Progressive Delivery, Chaos Engineering etc.) at a different advancement level.

Level 1 - Extended PoC

Provide a good enough starting point for experimentation.

The sooner the platform is available, the better. Start with rudimentary features to enable first applications to leverage the speed and flexibility of Kubernetes.
Use this level only as a starting point to experiment and test the possibilities. Be prepared for a rather reactive approach to problems and manual actions to fix them. The time will come for more advanced means of improvement.

Requirements

- ☑ A budget for the work to set up the platform and to deploy first applications
- ☑ Access to the cloud with Kubernetes service or available on-prem resources
- ☑ A team with skills for setting up the platform (greater for on-prem environments)

For whom

- ➔ Every organization starting the journey (non-prod!)

Benefits

- ★ Enables developers to learn and use Kubernetes and containers
- ★ Enables more advanced developers to leverage already built and used container images in a scale (previously locally)
- ★ Enables operations team to learn and discover how to create more mature platform with the available infrastructure (cloud providers or on-prem)
- ★ Allows for flexibility of adding more advanced options (i.e. additional products or services enhancing the platform features) before running real production workloads
- ★ Development and operations teams gain more experience

Do

- + Choose fast paths, even if they are now considered imperfect
- + Find and encourage technology enthusiasts to participate
- + Run your applications to align the platform use to the context

Don't

- - Implement more advanced features (e.g. RBAC, GitOps, Persistent Volumes)
- - Focus on scalability or security
- - Automate processes of building container images, delivery or platform provisioning

Level 2 – Basic

Discover useful features and explore further.

Let people continue to learn while the platform is improved and more features are added.
A small number of non-critical, stateless applications may run to prove the usability and benefits of using the new approach.

Requirements

- ☑ A small number of applications is ready for first production
- ☑ Basic roadmap with a set of applications to run on the platform
- ☑ Permission to learn and experiment with the delivery process
- ☑ Selected most important components (i.e. cloud provider, Kubernetes distribution, cluster architecture)
- ☑ Confirmed (or disconfirmed) usability of the new approach for the company - decision to move forward or stop and analyze

For whom

- ➔ Every organization that wants to build more mature and professional platform
- ➔ Companies (e.g. startups) choosing to make non-critical services available to clients

Benefits

- ★ Increased security rules to protect the applications and the platform
- ★ Increased visibility of platform performance and operations
- ★ Faster and more frequent deployments with rollbacks
- ★ Reduced response time and handle peak traffic with manual scaling
- ★ Enable developers to use of Kubernetes and containers
- ★ Ready for non-critical production workloads (small scale, small risk, stateless)

Do

- + Choose the necessary services or products (Kubernetes distribution, logging, identity)
- + Address doubts about platform capabilities with open communication
- + Implement basic security rules

Don’t

- - Force migration of existing, non-containerized applications
- - Optimize infrastructure costs (yet)
- - Standardize and unify delivery processes

Level 3 - Advanced

Learn and improve using reliable data.

The first semi-critical applications can run on in production. It's time to rely on the data collected on the platform to improve security, simplify troubleshooting, and reduce infrastructure costs.
At this level, more experience is required to leverage the potential of Kubernetes and Cloud Native projects.

Requirements

☑ Some apps work in prod to prove the readiness of platform
☑ Some standards have emerged (delivery)
☑ More skilled platform team(s)

For whom

➔ Preparing for more critical use
➔ More security required
➔ Availability becomes more critical

Benefits

★ More reliable platform (at least 99.9%)
★ Reduced risk of data leaks
★ Possible to scale both workloads and platform
★ Optimized utilization of resources
★ More people trust the provisioned platform

Do

+ Start implementing best practices in code (security rules, delivery pipelines, etc.)
+ Create a dedicated team for managing the platform
+ Improve platform capabilities based on data

Don't

- Force GitOps for all processes
- Delegate platform security to a dedicated team
- Run stateful applications on the platform (yet)

Level 4 - Professional

Manage platform with an "Everything as Code" approach.

The platform is now fully automated and improvements are being implemented in code (GitOps).
More proactive and automated measures are being used to improve application reliability, scalability and address security threats.

Requirements

- ☑ Large scale to justify higher costs
- ☑ Dedicated platform team or teams to manage the platform

For whom

- ➔ Large scale apps
- ➔ Stateful services
- ➔ Compliance standards requirements (e.g. GDPR, PCI DSS, HIPAA)

Benefits

- ★ More reliable platform (at least 99.99%)
- ★ Rapid delivery of more secure and reliable apps
- ★ Faster and optimized scaling capabilities
- ★ Infrastructure costs under control and available for optimization
- ★ Significantly reduced risks of data leaks and break-ins
- ★ Platform viewed internally as an essential service

Do

- + Start treating the platform as an internal product
- + Tighten platform security rules
- + Improve platform capabilities based on data

Don't

- - Announce the platform's SLA (yet)
- - Stop people from testing and experimenting (in safe environments)
- - Rely on a single infrastructure provider (or datacenter)

Level 5 - Expert

Remove bottlenecks and improve continuously.

The platform is ready for the most critical applications. It is continuously improved and offered as a key in-house product.

Requirements

☑ Organization's strategy to include continuous development and maintenance of the platform (software costs, people)

For whom

➔ Highest requirements for platform security, reliability and cost effectiveness

Benefits

★ Implemented Zero Trust Environment
★ Platform as a product with defined SLA
★ Capabilities to run any type of workloads (stateless, stateful, machine learning, serverless)
★ Detailed insight into the cost of operating the platform and application
★ High confidence in the platform's capabilities and reliability

Do

+ Prepare and announce the platform's SLA
+ Receive the official confirmation of platform compliance with security standards
+ Encourage people to run all their workloads on the platform

Don't

- Stop improving the platform

Contact

💡 Need more details?

↗ Need help getting to the next level?

Tomasz Cholewa
DevOps Architect, Trainer, Consultant

Feel free to reach out at tomasz@cloudowski.com to discuss details.