

4

OpenShift Personas and Skillsets

Now that you understand what an OpenShift cluster architecture looks like, what about understanding the people that work with it? I have seen in my career many adoptions of different software, platforms, and processes, some of them running smoothly and others being challenging. What are the differences between them that define good or bad adoption? The main difference is the people – that is, how people are treated in the process is crucial to determining good adoption.

We strongly believe that you need to care about people as much as you care about the platform itself. Because of that, we decided to reserve some pages of this book to discuss the main personas and skills usually needed to perform activities well.

Therefore, you will find in this chapter the following:

- Some of the personas that are related to OpenShift
- The key responsibilities and duties that are usually required for those personas
- The challenges they usually face
- What they usually expect to see in a container platform
- A skills matrix containing the main abilities that a resource needs to have

The following are the main topics covered in this chapter:

- Personas
- The skills matrix

Let's dive in!

Personas

We are going to discuss in this section the professional roles (**per-sonas**) that we have usually worked with in our career dealing with OpenShift and that, in general, are related to OpenShift on some level.

In the following figure, you will see some of the typical roles from an IT department:

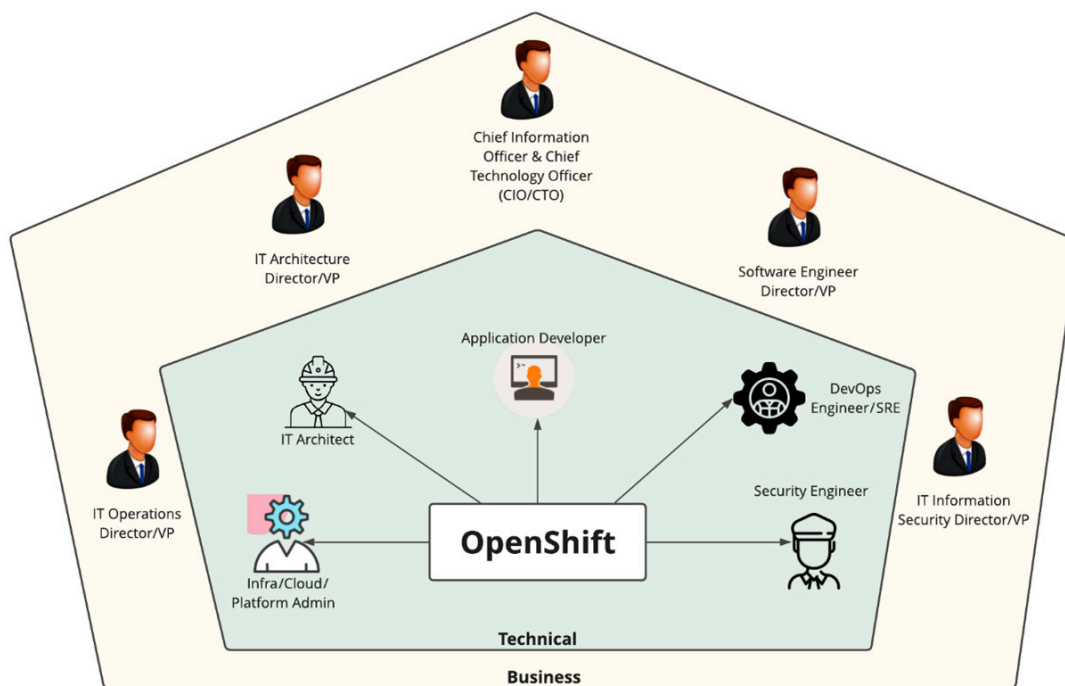


Figure 4.1 – Personas and OpenShift

Let's review some of the main technical roles and learn useful information that will allow us to better prepare an organization for OpenShift adoption. If you work in a coordinator role, take this opportunity to learn what you should expect from your technical team; if you are an individual contributor, learn what the activities and skills are that you will probably need to handle on a daily basis.

A system/cloud/platform administrator

A system, cloud, or platform administrator is an individual contributor who is responsible for maintaining the underlying infrastructure of the OpenShift clusters and also the clusters themselves. Depending on the scale of an organization, there are multiple specializations in this role, such as network, operating system, and storage administration.

Let's look at some of the duties they need to perform and the challenges they face while doing so.

Key responsibilities and duties

- Support the network, storage, and virtualization layers and the underlying infrastructure.
- OpenShift cluster administrative operations, such as scale nodes, upgrades, and granting permissions to users and groups.
- Configure OpenShift namespaces, network policies, resource quotas, node selectors, and so on.
- Platform capacity management.
- Cloud cost estimation and reporting.
- Open support tickets with vendors to get assistance to deal with issues related to the clusters' underlying infrastructure and/or OpenShift.
- Monitor the platform performance and stability.
- Analyze and report incidents proactively when noticing possible issues.
- Respond to users about incidents.
- Collect logs to investigate issues.
- Report metrics about operations, such as the number of incidents, **Mean Time to Recovery (MTTR)**, and **Mean Time Between Failures (MTBF)**.

The challenges they face (usually)

- **Standards:** As clusters grow, it becomes increasingly difficult to manage them mainly due to the lack of standards to use.
- **Observability:** Organizations that struggle to establish comprehensive and efficient application and infrastructure monitoring are not uncommon, unfortunately. Inefficient monitoring tools and practices make it much more difficult to make the right decisions. Administrators have difficulties in establishing tools and practices to observe environments properly.
- **Capacity planning and management:** Cluster administrators often have challenges with correctly managing the cluster's capacity. It is not an unusual situation where clusters run out of resources due to a lack of monitoring and capacity planning.
- **Cloud cost:** As cloud usage grows, keeping costs under control also becomes more difficult.
- **Supporting multiple cloud environments:** Administrators often need to deal with and support multiple cloud environments, which makes administration and management much more challenging.
- **Logging management:** Some companies struggle to define and maintain an efficient logging platform for containers, which impacts issue analysis, root-cause detection of a problem, and so on.
- **Support tickets:** It is common to have difficulties managing support tickets from vendors, sometimes due to lack of knowledge of how to collect logs and other things, such as insufficient information for a vendor to review, or vendor delay.

What do they want to see on a container platform?

- A mature enterprise container orchestrator platform to host the applications developed by the development teams.
- Vendors that can provide a great support experience and establish themselves as trusted advisor partners.
- A platform that is compatible with different infrastructure and/or cloud vendors.
- If there is a monitoring and/or logging tool in place, the platform needs to be able to integrate with it. If there isn't, the platform

needs to have an out-of-the-box logging and monitoring tool.

- A platform with available GitOps tools, such as Argo CD.
- Monitoring tools and dashboards that can assist in the capacity planning and management of the clusters.
- A platform with a high level of automation out of the box and/or can easily be automated using tools such as Ansible or Terraform.
- A complete product documentation and knowledge base available.
- Great training that is available from vendors or partners.

The next persona that usually has a tight relationship with OpenShift is the IT architect. Let's take a look at it closely now.

IT architect

An IT architect, in the context of OpenShift, is an individual contributor who is responsible for designing the high-level architecture and definitions for one or more clusters. Architects collect business requirements and understand issues, problems, and other references to define a solution that best addresses them. There are several different architect roles, which change from company to company, but some usual ones are the following:

- **Application architects:** Design and supervise the development and release of applications and related technical tasks.
- **Solution architects:** In general, architects with strong technical but also management knowledge who lead entire projects.
- **Enterprise architects:** An architect who works with the business to define the whole IT strategy for the company.
- **Infrastructure/cloud architects:** Design and supervise projects on cloud providers or on-premises hardware and infrastructure.

The architect role described as follows is closer to an infrastructure/cloud architect, who generally will work much closer with OpenShift than the others.

Key responsibilities and duties

- Work with the business to get the requirements for a container orchestration platform.
- Plan and design the cluster architecture, considering the business requirements and best practices such as high availability, security, and other important aspects for any critical platform.
- Estimate sizes to handle the initial required capacity.
- Estimate the effort to deploy and support the platform.
- Estimate the initial and ongoing cost of the platform.
- Assist the director and others at the VP level on the platform-buying decision.
- Design a monitoring and logging solution for the platform.
- Define security requirements for the platform.
- Define the ongoing policies and process requirements, such as backup and disaster recovery.
- Assist the DevOps engineer to define **Continuous Integration/Continuous Deployment (CI/CD)** processes and pipelines using the container platform.
- Hand over the decisions and specifications to the development and operations teams. Assist in deployment and platform usage.

The challenges they face (usually)

- **Knowledge:** I have seen many times IT architects that were pushed to design architecture for an OpenShift cluster without knowing sometimes even the basic concepts about it. They are used to designing architectures for applications in other less complex systems and not for a platform such as Kubernetes or OpenShift, in which you need to deal with a wide range of aspects such as virtualization, storage, and network.
- **Lack of sponsorship:** Architects usually need to get information and engage people from different teams inside an organization to understand and define some aspects of the architecture – for example, which storage options they have available to use with the plat-

form, which network characteristics will be used, and so on. When they don't have proper sponsorship to engage the right people, the architecture design usually is time-consuming and frustrating.

- **Requirement definitions:** It is not uncommon for organizations to not have a clear definition of what the requirements for the platform are. Some companies don't even know what they want to achieve with the platform and decide to adopt the technology just because some competitors are using it, or other similar reasons. That makes the architectural definition a much harder task, as it is difficult to be assertive and accurate with the decisions that need to be made in the process.
- **Vendor assistance:** Architects need a lot of assistance from the vendor during the platform's architectural design. Engaging with the vendor sometimes is challenging, due to a lack of availability, a delay in replying to information, and so on.

What do they want to see on a container platform?

- A mature enterprise container orchestrator platform to host the applications developed by the development teams
- Vendors that can meet service-level agreements for support
- Trusted advisor vendors that can assist with information, demonstrations, and proof of concepts
- A platform that is compatible with the infrastructure and/or cloud vendors defined by the business
- A platform that can integrate with existing tools, such as **Application Performance Management (APM)**, monitoring, and logging
- A comprehensive platform that includes monitoring, logging, CI/CD, service mesh, and serverless tools.
- GitOps tools available out of the box
- An ecosystem that includes IDE plugins, samples, and other development tools
- A platform with a high level of automation out of the box and/or can easily be automated using tools such as Ansible or Terraform

- A central management solution for multiple clusters
- A complete product documentation and knowledge base available
- Great training that is available from vendors or partners

A really important piece of this puzzle is the application developer; without them, there is no application to host on any platform! Let's take a look at the role now.

Application developer

A developer is an individual contributor who uses a container platform to build and deploy application code. In some companies, the application developer has direct access to the container platform to perform basic actions on it, such as deployment and configuration of the application in development and/or **Quality Assurance (QA)** environments; in other companies, they only have access to a CI/CD system to run a pipeline to check and deploy code automatically.

Key responsibilities and duties

- Understand functional requirements, use cases, or user stories to develop and test code.
- Develop code using best practices such as **Test-Driven Development (TDD)**, CI, Infrastructure as Code, and GitOps.
- Commit code frequently.
- Develop and run unit tests.
- Build the container application.
- Support QA teams to test an application.
- Bug-fixing of production issues.

The challenges they face (usually)

- **Knowledge:** Usually, organizations don't give the appropriate concern to a developer's platform knowledge. They are well educated with code languages, frameworks, IDEs, and related subjects but

not about the platform that will host the application – in this case, OpenShift. That is a frequent cause of rework in order to adapt the code to specific platform requirements.

- **Proper permissions:** It is not uncommon that developers don't have proper permission with a container platform to perform their duties. This impacts their performance and generates frustration.
- **Proper processes and tools:** Some developers don't have the proper tools and processes available to perform activities by themselves. Usually, they depend on other teams to get the proper IDE and runtimes installed, update the CI/CD pipeline to reflect a new change, and so on.
- **Logging management:** In some companies, developers are not allowed to have direct access to logs; they need to request it from another team, and sometimes, that takes a long time, impacting bug-fixing issues.

What do they want to see in a container platform?

- Plugins for the most popular IDE
- A desktop development environment with tools to build and test containerized applications
- Comprehensive samples with a wide range of software development stacks
- Container base images with different supported runtimes (such as Java, Spring Boot, Node.js, and Quarkus)
- Seamless integration with the CI/CD tool
- The UI developer console, in which they can see the application logs, monitoring, and configurations
- A complete product documentation and knowledge base available
- Great training that is available from vendors or partners
- A platform with proper permissions set that allows them to perform their duties

Some more recent roles are also closely related with OpenShift such as the DevOps engineer and the **Site Reliability Engineer (SRE)**. In the

following section, you will learn more about them.

The DevOps engineer/SRE

A DevOps engineer is usually the individual contributor responsible for developing and maintaining CI/CD pipelines to support development teams. They also analyze possible issues that occur with application build and deployments, using the CI/CD and container platform tools. This role can bring a lot of discussions, as DevOps itself is a culture and a set of practices to bring infrastructure and development together to be more agile and efficient, rather than a job role.

Similarly, an SRE is usually the one that brings some software engineering aspects to infrastructure to automate repetitive operational tasks and enables scalable and reliable environments.

DevOps and SRE are tightly related – an SRE can be represented as one specific implementation of DevOps practices, usually focusing more on infrastructure and operational problems and implementations, while the DevOps engineer usually works more with the CI/CD process, supporting development teams to deliver code more efficiently.

We have decided to conflate both roles into one persona but adapt it to your real-world environment, and consider the following information to see which best fits your role in your organization.

Key responsibilities and duties

- Understand CI/CD processes and develop pipelines accordingly.
- The continuous enhancement of CI/CD pipelines that support the development teams.
- Define, build, and deploy standards for containers along with CI/CD pipelines.
- Deploy and support tools used with CI/CD processes, such as Container Registry and the Git versioning tool.

- Automate processes related to CI/CD (such as the DNS and load balancer configurations).
- Support development teams regarding questions and issues concerning CI/CD pipelines.
- Automate repetitive operational tasks such as server provisioning and network configurations.
- Define and implement observability of services.
- Capacity planning.
- Plan and run chaos testing.

The challenges they face (usually)

- **Knowledge:** DevOps engineers and SREs are also usually well educated with automation-related things but not always with the container platform and its underlying infrastructure.
- **Proper processes and tools:** The best tools to develop a desired CI/CD, testing, and automation are not always available. In these situations, they are usually pushed to use workarounds, develop things with much more effort, and so on, which can be really frustrating!
- **Platform limitations:** These professionals often need to automate processes with legacy systems that don't have APIs, which makes automation development much more difficult.

What do they want to see in a container platform?

- Easy integration with existing CI/CD tools
- A UI console in which they can see the build and deployment logs and check the deployment objects
- A high level of automation out of the box
- A CLI and API that helps to automate manual steps
- Plugins for CI/CD tools that facilitate pipeline development
- An easy way to deploy an application over multiple clusters
- Out-of-the-box observability or one that can be easily plugged into an existing tool

- A platform that helps to implement perturbation models for chaos testing

Finally, we have the security engineer, who is always important in any process, even more so now in the age of ransomware and other real security concerns. Let's check the following section to see how they are related to OpenShift.

The security engineers

The security engineer is an individual contributor responsible for reviewing processes and tools that provide security for systems and data. They are always evaluating applications and systems to control environments from known threats, protect existing infrastructure to minimize vulnerabilities, and finally, detect and respond against attackers.

Key responsibilities and duties

- Evaluate possible platform vulnerabilities and create strategies to mitigate them.
- Review CI/CD pipelines and add security gates to maximize security.
- Establish a container image security scanning tool to detect any known vulnerability within images.
- Define security policies to be used with all clusters.
- Ensure that clusters are compliant with defined policies.

The challenges they face (usually)

- **Knowledge:** The security engineer is no different from the previous roles. They usually have real concerns with OpenShift, mostly because they are not well educated with it and, as such, don't understand how security is implemented in it.

- **Secrets management:** Management of passwords and sensitive data is still a challenge on Kubernetes. There are several vault systems available in the market already, but many companies still don't have a well-established practice and the tools to support it, which makes this a point of concern for security teams.
- **Permissions management:** It is a real challenge to define the right permissions level to ensure a secure platform that doesn't lock down developers and other teams, impacting their productivity.
- **Vulnerability scan:** Security tools for containers and Kubernetes are fairly recent, so most companies don't have any available, which can cause real nightmares for security engineers!





What do they want to see in a container platform?

- A platform that uses the most secure standards for container orchestration, such as Linux namespaces, **cgroups**, Linux capabilities, **seccomp**, and SELinux
- A tool that allows them to easily define and enforce security policies among several clusters
- A multi-tenant secure-capable platform that enables network isolation between different workloads
- Data encryption at rest
- A secure **Role-Based Access Control (RBAC)** platform available
- An audit log enabled and logging
- Image registry with a security scan

We have covered the different technical roles and their responsibilities, which will help you and your organization to be more prepared for OpenShift adoption. Finally, we will present a skills matrix that aims to help understand what the main skills required by each role are.

The skills matrix

In this section, you will find a skills matrix suggested for the roles listed in the preceding sections. The items listed here are a subset of technical skills that we have observed from experience as being important for a professional to know. Note that we are not covering soft skills in this book; this doesn't mean that we don't place any importance on soft skills – on the contrary, we do! – however, they are not the focus of this book. The skills matrix is classified in the following criteria:

	Basic knowledge – able to perform basic tasks independently
	Advanced level – has the skills to complete more complex tasks with little or no assistance
	Expert level – has the skills to complete complex tasks and also pass knowledge on
	Specialist level – expert level but also able to work on optimizations and solution design for complex problems around a topic, and seen as a subject matter expert by other colleagues and partners

Check out the skills in the following tables, which we have split into parts for better readability.

Architecture-, infrastructure-, and automation-related skills

Check the following table for the skills needed for architecture, infrastructure, and automation tasks:

Skill	Platform administrator	IT architect	Application developer	DevOps engineer/SRE	Security engineer
Incident and problem handling					
System and software architecture design/modeling	N/A		N/A		
Cloud service providers (AWS, Azure, GCP, and so on)					
Networks, operating systems, and storage					
Automation tools (Ansible, Terraform, and so on)					
Capacity planning					
























Development-, container-, and CI/CD-related skills

In the following table, you will find the skills related to development and CI/CD:

Container building tools (Docker Compose build, S2I, Buildah, and BuildConfig)					
Container runtime (Docker, Podman, and so on)					
CI/CD tools (Jenkins, GitLab CI, Tekton, and so on)					
Infrastructure as code/GitOps (Argo CD, Flux CD, and so on)					
Logging tools (ELK, EFK, Loki, and so on)					
Monitoring/APM tools (Prometheus, Grafana, Zabbix, AppDynamics, Dynatrace, Amazon CloudWatch, Azure Monitor, and so on)					
Container image registries (Quay, Harbor, AWS ECR, Azure CR, Docker Hub, Artifactory, Nexus, and so on)					
Programming languages used by the organization (Java, Python, and so on)					
Databases used by the organization (MySQL, MongoDB, and so on)					

OpenShift-related skills

Finally, see the following table for the skills required for OpenShift:

OpenShift installation (*)			N/A	N/A	
Authentication and authorization					
Container Network Interface for OpenShift/ K8s (OVN/Ingress/Egress)					
Storage for OpenShift/K8s (in-tree/CSI)					
OpenShift Day 2 (scaling nodes, upgrades, operators' deployment, and so on)					

(*) If administrators have assistance from a consulting partner to deploy the clusters. If they are required to deploy new clusters, then they should have at least an advanced level of OpenShift installation.

Summary

We have seen in this chapter the main personas related to OpenShift, their key responsibilities, the challenges they usually face, and what they look for in a container platform. Now, we also have an idea of the competencies an OpenShift professional needs to have. We expect it to help you prepare yourself for the tasks you will perform as an individual contributor or plan your team enablement and hiring policy accordingly, if you are at a management level.

This chapter concludes the first part of this book, reserved for concepts, architecture, and enablement. Now, go to the next chapter and start your journey on OpenShift cluster deployment. In that chapter, we will walk you through the steps to prepare the installation, deploy a cluster, and perform some post-deployment tasks.

Further reading

If you want to look at more information related to the concepts we covered in this chapter, check the following references:

- *What is a skills matrix and should I use one in 2021?:*
<https://www.skills-base.com/what-is-a-skills-matrix-and-should-i-use-one-in-2021>
- *Everything-as-Code:*
<https://openpracticelibrary.com/practice/everything-as-code/>
- *GitOps:* <https://openpracticelibrary.com/practice/gitops/>
- *How Full is My Cluster? Capacity Management and Monitoring on OpenShift:* <https://cloud.redhat.com/blog/full-cluster-capacity-management-monitoring-openshift>
- *A layered approach to container and Kubernetes security:*
<https://www.redhat.com/en/resources/layered-approach-container-kubernetes-security-whitepaper>
- *OpenShift Container Platform architecture:* <https://docs.openshift.com/container-platform/4.9/architecture/architecture.html>

Previous chapter

< [Chapter 3: Multi-Tenant Considerations](#)

Next chapter

[Part 2 – Leverage Enterprise Products with Red Hat OpenShift](#) >