# KLA Employee Training Program – Fursa Graduate Track

**Introduction**

This training program was developed for KLA employees who are graduates of the Fursa program. It provides them with essential DevOps knowledge, hands-on infrastructure skills, and advanced programming capabilities needed to succeed in their roles at KLA. The curriculum integrates practical assignments, real-world tools, and guided learning, focusing on topics such as CI/CD (Jenkins, Azure Pipelines), Infrastructure as Code (Terraform, Pulumi, Ansible), Linux and Windows system operations, Docker, and Cloud QA methodologies.

All training materials—including code samples, documentation, and exercises—will be uploaded to a dedicated GitHub repository shared with Vlad and Adham. Each topic folder will include:

* Practice files (e.g., Jenkinsfile, YAML, Dockerfile)
* Topic-specific README guides
* Defined tasks with expected outputs
* Directories for deliverables (docs/, scripts/, solutions/)

## Training Curriculum Table (Updated)

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Topic** | **Topic Description** | **Remarks** |
| 1 | Introduction to KLA and the Role | Introduction to the company, areas of activity, products, technologies, organization structure, and the employee's role |  |
| 2 | Jenkins Pipelines and Shared Libraries | Building pipelines in the polybot project, using parameters, writing shared code libraries |  |
| 3 | Azure Pipelines | Converting pipeline to Azure DevOps, working with YAML, stages, variables, approval gates |  |
| 4 | IaC – Introduction and Tools | Introduction to Infrastructure as Code and tools: Terraform, Pulumi, Vault, Consul, Artifactory |  |
| 5 | Terraform for Windows Projects | Writing code for Windows-based infrastructure using variables, outputs, and modules |  |
| 6 | Pulumi | Writing cloud infrastructure using Pulumi with Python |  |
| 7 | Artifactory | Managing build artifacts and Docker images, version control and cleanup policies |  |
| 8 | Vault and Consul (Optional) | Secret management and service discovery – basic overview of HashiCorp Vault and Consul |  |
| 9 | Ansible | Running processes on remote servers using Playbooks – according to Alon's guide |  |
| 10 | Packer | Creating code-based infrastructure images (e.g., AMI) |  |
| 11 | Windows Dockers | Building Windows-based containers, writing a Dockerfile for IIS and local deployment |  |
| 12 | Linux Administration | Core commands, permissions, log analysis, SSH, service management |  |
| 13 | Advanced Python – Multithreading and Async | Parallel and asynchronous programming: ThreadPoolExecutor, asyncio, performance comparison |  |
| 14 | Project Development Process + Test Plan | From idea to development – writing SRD, process documentation, building a test plan |  |
| 15 | Manual Testing – Cloud Infrastructure QA | Test planning, bug analysis, end-to-end and API scenarios for VPC systems (based on the assignment file) |  |

**Topic 7: Artifactory – Managing Build Artifacts and Docker Images**

**What you’ll learn:**

* Managing software packages, builds, and Docker images using **Artifactory**
* Automating artifact uploads and applying cleanup policies
* Understanding how Artifactory fits into the CI/CD pipeline

**Tasks:**

* Upload a file to Artifactory using a curl command
* Create and document a policy for removing old artifact versions
* Set up folder structures for organizing builds
* Integrate Artifactory into a build process (optional for advanced students)

**Deliverables:**

* curl command used and output log
* A configuration document describing version retention policies
* Markdown documentation with folder layout and upload instructions

**Links:**

* [Artifactory Quick Start](https://jfrog.com/artifactory/start-free/)
* [Artifactory Documentation](https://jfrog.com/help/r/jfrog-artifactory-documentation)

בהחלט. הנה תרגום של נספחים 8–10 מתוך תוכנית ההדרכה:

**Topic 8: Ansible – Remote Server Automation**

**What you’ll learn:**

* Running Playbooks to manage remote servers
* Writing tasks using modules and templates
* Working with inventories to define host groups

**Tasks:**

* Read Alon’s Ansible guide (link below)
* Run a Playbook that installs Apache on a remote server
* Add a Jinja2 template for a custom index.html page

**Deliverables:**

* playbook.yml for Apache setup
* inventory file specifying the target server
* Markdown documentation describing the output and process

**Link to Alon's guide:**  
<https://github.com/alonitac/Fursa25/tree/main/ansible_workdir>

**Topic 9: Packer – Creating Prebuilt Machine Images**

**What you’ll learn:**

* Creating reusable VM images (e.g., AMI, VHD)
* Using **Provisioners** and **Builders** to automate image creation
* Maintaining consistent environments through image cloning

**Tasks:**

* Write a packer.json file to build a Windows AMI
* Add a provisioner that installs software like Python
* Execute the image build process and document each stage

**Deliverables:**

* A complete and documented packer.json file
* CLI output logs from the image creation process
* Markdown documentation explaining each stage of the build

**Links:**

* [Packer Tutorials – HashiCorp](https://developer.hashicorp.com/packer/tutorials)

**Topic 10: Windows Dockers – Building Containers on Windows**

**What you’ll learn:**

* Building and running Windows-based containers
* Differences between Linux and Windows containers
* Working with Docker Desktop for Windows environments

**Tasks:**

* Write a Dockerfile configured to run IIS
* Expose port 80 and verify in a browser
* Push the container to Docker Hub

**Deliverables:**

* Dockerfile
* Output from docker build and docker run
* Screenshot showing the service running in a browser

**Links:**

* [Microsoft – Windows Containers](https://learn.microsoft.com/en-us/virtualization/windowscontainers/)
* [Video Tutorial – IIS on Docker](https://www.youtube.com/watch?v=z726_zDvMLk)

**Great. Here's an enhanced, advanced version of the Linux Administration topic, with a stronger emphasis on Bash scripting and automation—suitable for technical roles:**

**Topic 11 (Advanced): Linux Administration and Bash Scripting**

**What you’ll learn:**

* **Mastering core Linux administration tools**
* **Automating daily tasks using Bash scripts**
* **Parsing log files, managing processes, and user roles programmatically**
* **Writing idempotent scripts for system configuration**

**Advanced Concepts Covered:**

* **Cron jobs and scheduling scripts**
* **Error handling and logging in Bash**
* **Conditional execution and command chaining (&&, ||, set -e)**
* **Text processing tools: awk, sed, cut, xargs**
* **Basic use of expect scripts for automating interactive CLI tasks (optional)**

**Tasks:**

1. **Log File Monitoring Script:**
   * **Write a Bash script that scans /var/log/syslog or /var/log/auth.log**
   * **Filters out all lines with failed login attempts**
   * **Sends a summary email or writes a report to /tmp/login\_failures.txt**
2. **User Audit Script:**
   * **Script that lists all users with shell access**
   * **Verifies which ones logged in in the past 7 days**
   * **Outputs results in a table format using column or CSV**
3. **Service Health Check Script:**
   * **Check the status of key services (e.g., nginx, sshd, docker)**
   * **If a service is inactive, restart it and log the action**
   * **Use systemctl, tee, and logger**
4. **Scheduled Task:**
   * **Create a cron job that runs one of the above scripts every 15 minutes**
   * **Log output to a dated file in /var/log/custom/**

**Deliverables:**

* **monitor\_logins.sh**
* **user\_audit.sh**
* **healthcheck.sh**
* **crontab entry file**
* **Markdown file: description, usage instructions, and setup notes**
* **Bonus (optional): use expect to automate ssh into another machine and check disk space**

**Links:**

* [**Advanced Bash Guide**](https://tldp.org/LDP/abs/html/)
* [**Bash Best Practices**](https://google.github.io/styleguide/shellguide.html)
* [**Linux Command Line Crash Course**](https://linuxcommand.org/lc3_learning_the_shell.php)
* [**Explaining Expect**](https://www.thegeekstuff.com/2010/10/expect-examples/)

**Topic 12: Advanced Python – Multithreading and Async Programming**

**What you’ll learn:**

* Using ThreadPoolExecutor for parallelism
* Working with asyncio, coroutines, and await
* Comparing performance between synchronous and asynchronous code

**Tasks:**

* Write code that makes multiple API calls using ThreadPoolExecutor
* Implement an async version using aiohttp
* Compare and document runtime differences

**Deliverables:**

* multithread\_example.py – with multithreaded API calls
* async\_example.py – with async/await logic
* Markdown explanation comparing execution time and performance

**Links:**

* [Python Concurrency Overview – Real Python](https://realpython.com/python-concurrency/)
* [Asyncio Tutorial – Real Python](https://realpython.com/async-io-python/)

**Topic 13: Project Development Process + Writing a Test Plan**

**What you’ll learn:**

* Steps in the development process: idea → requirements → SRD → test planning
* Writing a Software Requirements Document (SRD)
* Creating a test plan with structured test types

**Tasks:**

* Choose a simple product idea (e.g., Book Management System)
* Write an SRD including requirements, system description, boundaries
* Write a Test Plan covering objectives, positive/negative cases, and boundary conditions

**Deliverables:**

* SRD.docx or .md file
* Test\_Plan.docx in IEEE format or Atlassian format
* Flowchart illustrating the development process

**Links:**

* [Test Plan Guide – Atlassian](https://www.atlassian.com/continuous-delivery/software-testing/test-plan)
* [Test Plan Template – Guru99](https://www.guru99.com/software-test-plan.html)

**Topic 14: Manual Testing – Cloud Infrastructure QA (Based on Assignment File)**

**What you’ll learn:**

* Designing test scenarios for cloud infrastructure systems
* Bug analysis and reporting
* End-to-End and API testing strategies

**Tasks:**

**Part 1 – Test Scenario Planning:**

* Write at least 10 VPC test cases covering:
  + Positive scenarios
  + Boundary tests (e.g., CIDR /28, /16)
  + Negative scenarios (e.g., invalid CIDR, overly long names)
* Use format: ID, Description, Steps, Expected Result, Category, Priority

**Part 2 – Bug Reporting:**

* Write 3 bug reports covering:
  + DNS resolution
  + CIDR validation
  + VPC creation failure
* Include: Description, Reproduction Steps, Actual vs Expected, Severity Rating

**Part 3 – End-to-End Testing:**

* Plan a complete scenario for VPC with 3 layers (load balancer, app, DB)
* Define validation points: connectivity, security, performance

**Part 4 – API Testing:**

* Create test cases for POST / GET / PUT / DELETE operations
* Include invalid input, permission checks, and stress testing

**Deliverables:**

* test\_cases.xlsx or .md file
* bug\_reports.md
* end\_to\_end\_plan.md
* api\_scenarios.json with explanations

**Links:**

* [API Testing – RESTful API Guide with Postman](https://www.youtube.com/watch?v=wEOLZq-7DYs)
* [Bug Report Template – Guru99](https://www.guru99.com/how-to-write-a-bug-report.html)
* [Postman Tutorial – Guru99](https://www.guru99.com/postman-tutorial.html)