**DS/CS553 - Machine Learning Development and Operations (MLOps)**

# CASE-STUDY 2

## Group 2: Ask the Greats

**Group Members:**

Andrew Keane

Jake Watson

Deep Suchak

**GitHub Repo**:

<https://github.com/jakewatson91/cs553-group2>

## Virtual Machine Setup Process

**SSH Access Setup**

We initially set up SSH access to the virtual machines by generating private and public keys on our local machines and storing them in our .ssh directories. We saved the public keys for each group member in the authorized\_keys file for the student-admin account of the virtual machine, securing it by removing the temporary access key.

By doing this we were able to log in using the command: ssh -i $HOME/.ssh/<individual\_keys> -p 22002 -o StrictHostKeyChecking=no student-admin@paffenroth-23.dyn.wpi.edu

Each of these steps is built into the mlops\_keys.sh script, which automates the key generation process.

**Environment Configuration**

The environment configuration is handled via the mlops\_product.sh script. Key steps include:

* **Installation:** The python3-venv package is installed using `sudo apt install -qq -y python3-venv`. The -y flag automatically answers “yes” during the installation process.
* **Creation:** A virtual environment named “venv” is created in the product's directory using python3 -m venv venv.
* **Activation and Installation:** The virtual environment is activated, and dependencies from requirements.txt are installed to ensure the isolation of libraries.
* **Application Startup:** The application is launched using the virtual environment's Python interpreter, with output directed to log.txt for monitoring.

## Deployment Process

The deployment process is divided between two scripts: mlops\_keys.sh and mlops\_product.sh

**mlops\_keys.sh**

This script ensures all team members can regain access to the virtual machines without manual intervention by restoring public SSH keys in the event of a system reset.

After establishing access, we configured the necessary software and environment variables for the applications. A key challenge was managing user permissions, which we addressed by creating individual accounts with secure access to their directories. Additionally, we configured network settings and port forwarding to ensure secure communication between services during deployment.

**mlops\_product.sh**  
Once SSH keys have been set up, the mlops\_product.sh script deploys both products on the virtual machines.

* **Cloning Git repository:** the script creates a dedicated directory for the GitHub repository, removing any existing versions to avoid conflicts. It then clones the latest code from GitHub and securely transfers the files to the remote server using SSH keys, addressing authentication challenges effectively.
* **Virtual Environment set up:** After verifying the files' presence on the server, the script installs a Python virtual environment and the necessary dependencies from the requirements.txt file.
* **Launching applications:** The applications are launched using nohup, allowing them to run in the background even if the SSH session is closed.

## Automated Recovery

The automated recovery process (in the mlops\_keys.sh) detects when the server has been wiped with this block of code:  
testhost=`ssh -i $MLOPSPERSKEY -p $MLOPSPORT $MLOPSHOST hostname 2> /dev/null`  
if [ "$?" != 0 ]

An attempt is then made to SSH to the host using a personal key. If this fails, the assumptions in the script are that the host has been wiped and refreshed, reverting to the default student-admin\_key. In this case, mlops\_keys.sh restores accounts and the personal (public) keys are reestablished on the server.

**Automation with crontab:** The recovery automation is triggered with the following crontab command running on a local laptop:  
\* \* \* \* \* bin/mlops\_sync.sh >> bin/mlops\_sync.log 2>&1

This command triggers the key restoration process, as well as the environment and product setup within one minute of the server coming back up in a refreshed state.

**Effectiveness:** The scripts are an effective way to automatically restore secure access, and a working product on the virtual machine with minimal downtime and without any manual intervention. Permissions for directories and SSH keys are set securely, following best practices.

**Limitations:** Crontab automation is limited as it requires the personal laptop to be on, connected to the Internet, and connected to the WPI network (either in person or via VPN). Within the scope of this course and without access to persistent hardware, this was the closest approximation we could perform.

While the key setup is relatively fast (~30 seconds), the product setup takes up to 23 minutes. The majority of this is spent installing the required packages, which takes 5.3GB of space on the virtual machine.

The script relies on some hardcoded user accounts and key paths, which can be less reliable and require manual changes for updates. Missing or misnamed public keys can cause access failures for users.

**Potential Improvements:**

**Error Handling:** Adding error checks after each command would improve feedback and handle failures (e.g., user creation or permission issues) more gracefully.

**Dynamic User Management:** Integrating dynamic user lists from a configuration file or database would enhance adaptability to team changes.

## Additional Insights, Challenges, and Future Improvements

**Challenges Faced:**

* **Product Deployment:** Ensuring the Python virtual environment and dependencies were correctly set up required precise handling, especially during virtual environment creation and package installation.
* **Server/VM Access:** Accessing the group server/VM remotely via the WPI VPN was periodically slow or unreliable with network drops. That sometimes leaves the server in a bad state when a perfect connection would not have caused any issues.
* **Installation/Recovery Time:** While most parts of the server recovery (e.g., replace SSH keys, download and copy over product) took one minute to complete, one part required around significantly more time. Spinning up the virtual environment includes performing the pip installs of the requirement tree defined in requirements.txt. The downloading and installing of those required dependencies requires about 30 minutes of time.

**Future Improvements:**

* **Monitoring:** Implement logging and monitoring tools (e.g., Prometheus, Grafana) to track application performance and system health in real-time.
* **Server provisioning:** Automate SSH key management and server provisioning to improve the recovery process in case of system wipes or key failures.
* **Recovery:** Currently when the system detects any problems with the product, the entire application is rebuilt from scratch. More fine-grained monitoring would help improve the efficiency of this by rebuilding only what is necessary.
* **Automation:** Create a Github action so that any changes to product code get pushed to our machine.

## Conclusion

In the wise words of Prof. Paffenroth, “Computers are disposable, computer configuration is forever.”

Setting up a secure and performant machine learning application is a complex process involving many steps. Automating these steps can significantly reduce downtime, errors, and security vulnerabilities in the event that the physical machine experiences issues or resets. It also saves hours of time for developers, allowing them to focus their efforts on building cool new products.