The Team for T1

📝 Codey Funston - Bak architecture and reverse proxy

💾 Susmitha Gogireddy - Bak script automation

📈 Kosta Thomson - Monitoring dashboards and alerting

💾 Shalom Ingabo - Bak API development

📈 Yuvin Dediwalage - Monitoring dashboards

Projects Motivation

The initial goal for the infrastructure team this semester was the following:

> To ensure the continual operation of company projects without disturbance or deletion from their underlying infrastructure, across on-premise virtual machine/s and various big-cloud services. And to allow for real-time insight into the state of all major components of the system._

1. Continual Operation Without Disturbance

Progress must move quickly throughout the company due to length of each semester. It would be bad for progress as well as relationships if our team kept requiring downtime or forcing migration when it may not be necessary.

2. Real-time Insight of State

Things will always go wrong so it is important to know as soon as they do.

3. No significant infrastructure resource changes

We will try to leave all virtual machines and cloud services as they are and avoid overreaching into the work of other teams.

4. Leave previous cyber security research and testing

This will remain valid since we are only planning to add supporting services, change management, and monitoring, as opposed to new network design or major server setup.

Overview of Projects

Automated Docker container version control

This project was partially completed with full design documentation, API specification, and intial testing with BASH and Python approaches for the core logic. As such, it would be very easy to continue on if it interested the next semester's team. The source files are located at redback-cyber/T1_2025/infra/backups and this includes the initial proof of concept, version 1, and documentation in the form of control flow diagrams and READMEs.

Some notes on key parts:

- Rsync was chosen as the core tool for the backend since it allows for snapshots not just backups. Snapshots only include raw copies of what has changed between versions, any unchanged data is represented by hard-links and so take up approximately no extra space. This removes the need for deletion of old copies and often complex rotations. If you have not done much Unix-based work a hard link is like a pointer to the data held by a file and so even if the file is deleted or moved the chunk of data is still there and can be pointed to. This is different to a soft link which is less like a pointer and more like an alias as it simply points to another filename.

- The design of the tool is split into a CLI, central controller/server, and a database. This is so that teams can use a simple CLI to register, update, remove, or version their container data without having to have any system administration experience - a successful infrastructure team will be needed less the more successful they are (in my personal opinion). The controller is what actually does the snapshots when configuration changes from CLI calls to the API. The backend doesn't directly run the scripts in its process execution, instead it dynamically updates a central Crontab.

So why...?

The motivation for the project was initially from a generic requirement for backups. Once we realized that we could no easily perform off site backups due to cost the focus was switched to version control. Without the proposed tool you can not rollback to previous states for Docker containers with persistent volumes without affecting all other containers. This tool goes into the Docker directory at `/var/lib/docker/volumes/` and takes direct snapshots/deletes and replaces the directories containing the persistent data. If this tool is chosen to be continued and implemented it means that teams can either iterate fast with their deployments and rollback with changes or they can accidentally wipe important data and not worry.

How to continue:

1. Get a working version of the script that takes a snapshot and rolls back a container volume.
2. Set up a routine that the controller executes to update Crontab.
3. Finish off CLI or optionally get users to use YAML/JSON configuration files instead which feed into API calls to the controller.

Monitoring and Logging

This project was intended to centralize all the metrics and logs for each project into structured dashboards. Logging was delayed for most of the semester and partially dropped near the end. Metrics monitoring implemented this semester included multiple dashboards for VM specific health and container health for each instance running on the Docker engine. The configuration for the deployment is in the infra-team directory in the cyber repo at .../monitoring. The Grafana dashboard is hosted on the VM at port 4444 and has a persistent volume attached so configuration will still be there. Prometheus was not deployed in production. If you decide to continue with this it would be best to consult blue team first as they do a lot of monitoring and log correlation for security.

Application Access

This project was very cool to work on and get a working test with basic functionality. Currently, the VM is accessed via redback.it.deakin.edu.au and individual projects/services are accessed by using ports. There is also HTTP and HTTPS access enabled. Both these things are unprofessional and harder to use than clearly laid out URl endpoints. The test was performed using CNAME records which is where more detailed URLS point to the same A record URL - note that Deakin IT cannot do this it instead has to be X.it.deakin.edu.au. The requests all come into the VM through the reverse proxy - Nginx - with the same destination IP but different host URL paths which Nginx filters out and redirects to the appropriate service.

To continue on it is best to speak to Robin Spoerl if he is continuing it one in T2 2025, other wise what is needed is set up for serving applications from sub paths since some don't like that ie Grafana. Also proper TLS is required, the demo I did used a basic openssl generated one, ideally Nginx would include single sign on with Deakin credentials and centralized access control for each application.

Virtual Machine Admin

File locations from this semester

All files created are under `/home/codey/infra-team`. Feel free to move this. All services/apps are run through Docker with docker-compose.yaml files only, there are no binaries to find from this semester.

Adding junior team members to the VM

It is best to give junior team members non-sudo access to the VM. The only root-like power that should be granted is Docker access. This command adds a user *David*, creates a home directory for them, and adds them to the Docker group so that they can run Docker commands without sudo:

```
sudo useradd -m -G docker -s /bin/bash david
```

Upgrading Docker package

Upgrading Docker will restart the engine. Any containers without a restart policy will exit so please check running containers before upgrading.

OS Upgrade

The VM was upgraded to Ubuntu 22.04 LTS during the last week of the semester. The end of life is April 2027. Before upgrading ask the Deakin IT team for a new snapshot and have each project lead test their applications before deleting the snapshot - unless the snapshot can be kept.