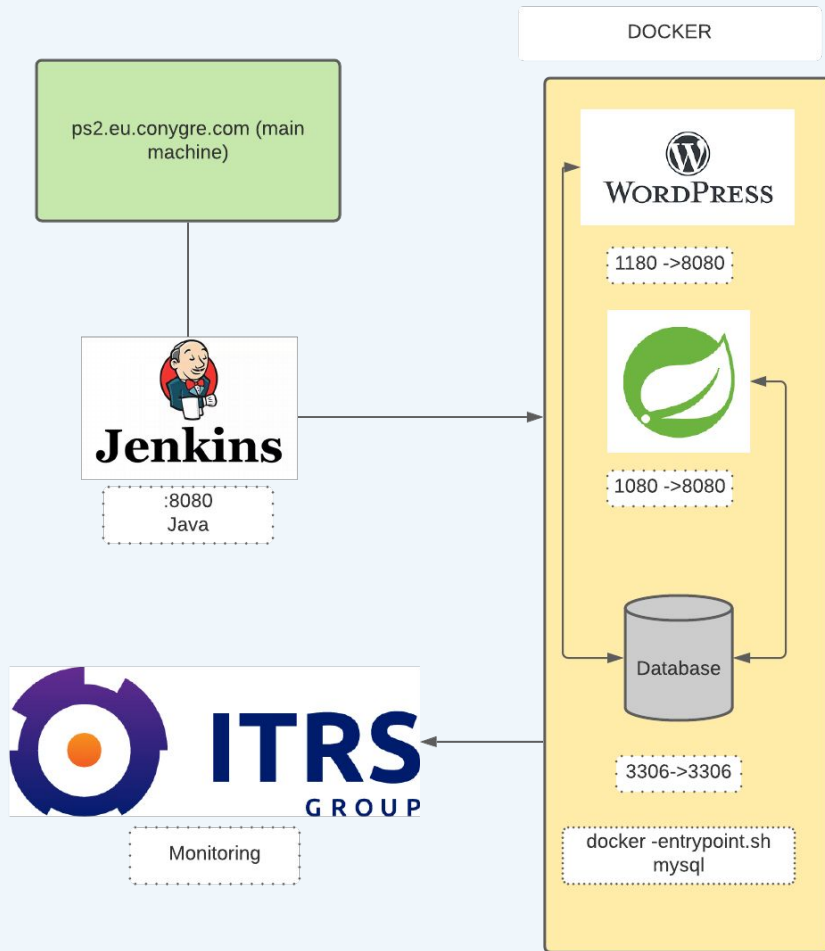




## Team 2

Sadiq Alibhai, Katrina Buchanan, Claire Maguire & Ross McCully



## Architecture

Our main machine, ps2.eu.conygre.com has Jenkins on Port 8080.

Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying. It facilitates continuous integration and continuous delivery.

We have our applications Wordpress and Petclinic which connect to the MySQL Database in the Docker container within Jenkins.

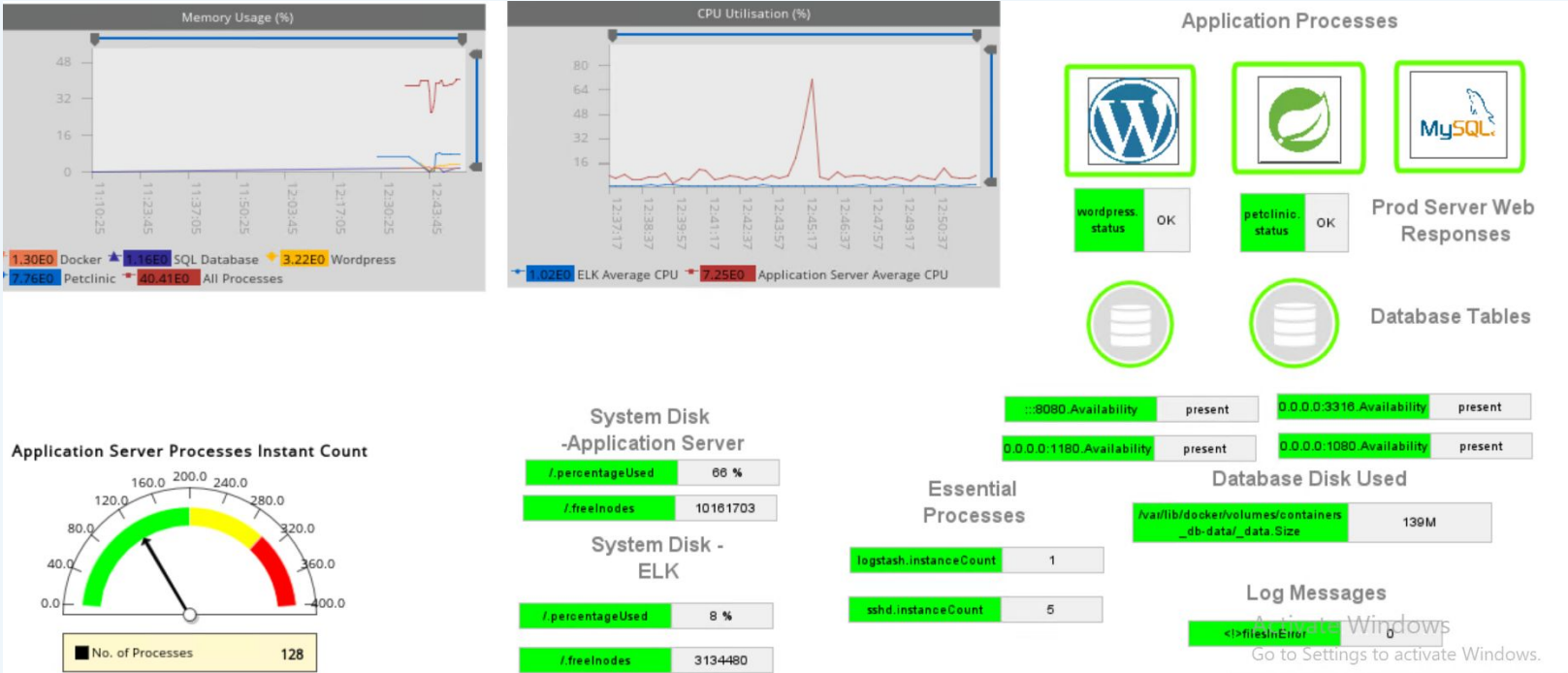
Wordpress is accessible on the main machine through port 1180 which is forwarded to a virtual machine port 8080.

Petclinic is accessible on the main machine through port 1080 which is forwarded to a virtual machine port 8080.

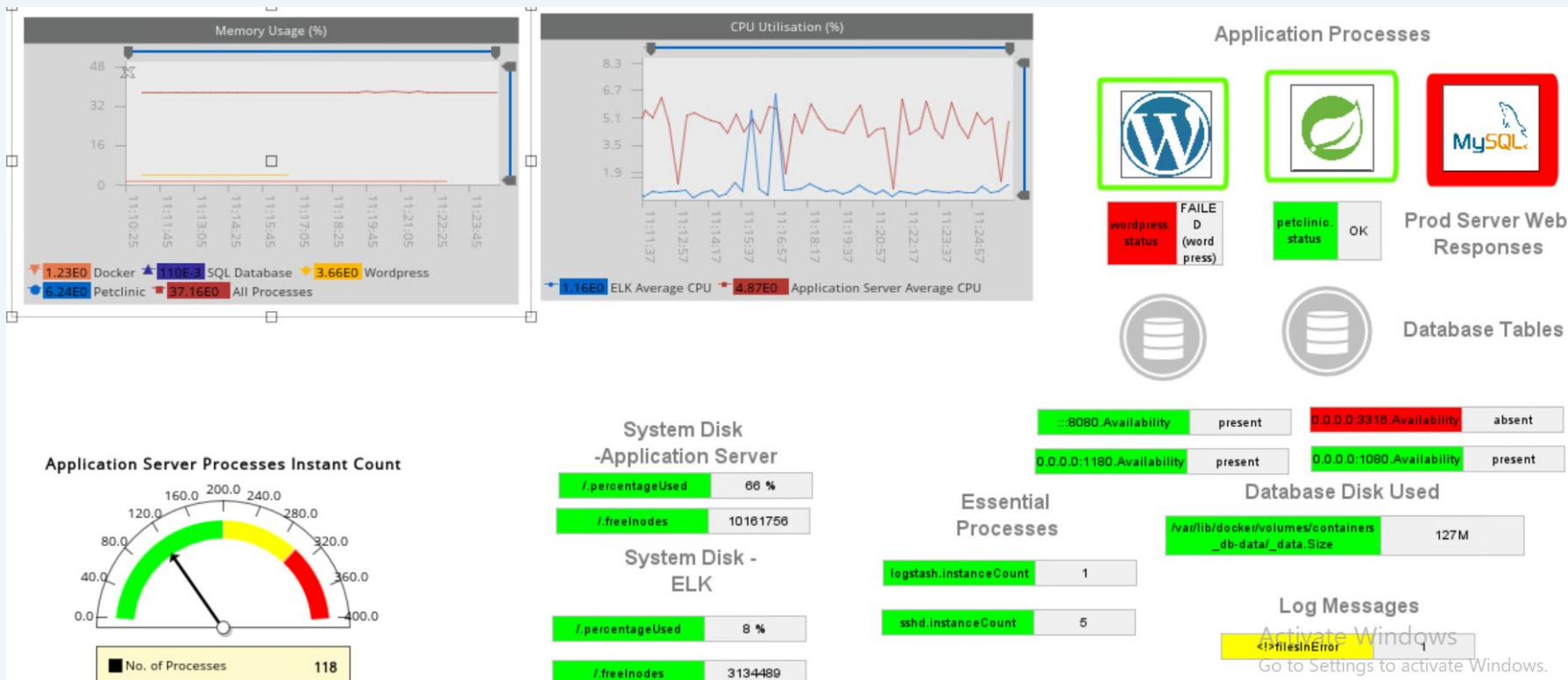
ITRS is our monitoring tool, which we will go into more detail about throughout the presentation.

# Monitoring

For monitoring our application, we created an active dashboard which allows us to view the status of our processes and applications. We shared the dashboard .adb file in the Bitbucket repository This will allow the team to see when a process is down to allow early detection and prevention to causing further issues.



Below is the Active Dashboard after the Database has been taken down. We can see that the MySQL Application Processes has turned red, the database tables have turned grey, this is because they do not exist without the database. We can also see our Port :3316 connection is absent which is the port the database listens on.



## Incident Management

- Incidents could be discovered from our monitoring systems including ITRS through the active dashboard, other employees/team members or clients.
- ITRS provides a level of concern including Warning and Critical. This could be used to help determine the incident severity level.


### Incident Management Process






# Jenkins

Jenkins helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery.

# Jenkins dashboard

 Jenkins

search ?

 1  2 Administrator  log out

Dashboard

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue

No builds in the queue.

Build Executor Status

















1 Idle


2 Idle

3 Idle

4 Idle

All Build Maintenance Start Stop +

S	W	Name ↓	Last Success	Last Failure	Last Duration	
✓	⚙	Build Projects	1 day 0 hr - #32	7 days 3 hr - #18	0.52 sec	
⋮	⚙	Compile All	N/A	N/A	N/A	
✓	☁	Database Backup PetClinic	7 min 8 sec - #41	2 hr 7 min - #40	1.9 sec	
✓	☁	Database backup wordpress	1.7 sec - #14	1 hr 5 min - #13	1.4 sec	
✓	⚙	Docker up all	1 day 5 hr - #3	N/A	9.1 sec	
✓	⚙	Launch Database	52 min - #16	7 days 2 hr - #5	4.7 sec	
✓	⚙	Launch Monitoring	2 mo 19 days - #1	N/A	3.5 sec	
✓	⚙	Launch PetClinic App	52 min - #44	7 days 2 hr - #29	6.4 sec	
✓	⚙	Launch Wordpress	52 min - #40	N/A	6.3 sec	
✓	☁	Restore Database petclinic	21 hr - #5	23 hr - #4	3.4 sec	
✓	☁	Restore Database wordpress	21 hr - #113	1 day 5 hr - #109	2.8 sec	
✓	⚙	StartDB, Wordpress, PetClinic	1 day 3 hr - #9	N/A	2 min 7 sec	
✓	☁	Stop PetClinic	1 day 21 hr - #16	7 days 1 hr - #12	12 sec	
✓	⚙	Stop Wordpress	1 day 5 hr - #5	N/A	3.3 sec	
✓	⚙	StopDatabase	2 hr 27 min - #8	N/A	5 sec	
✓	☁	StopDB, Wordpress, PetShop	6 days 22 hr - #6	6 days 22 hr - #3	17 sec	

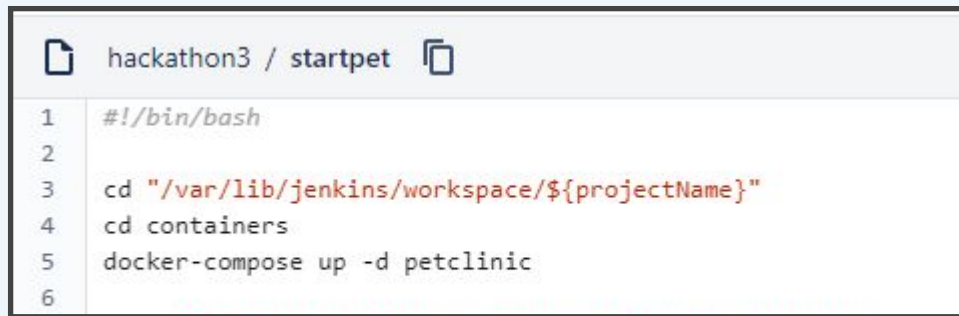
 add description

Activate Windows

# Source code management

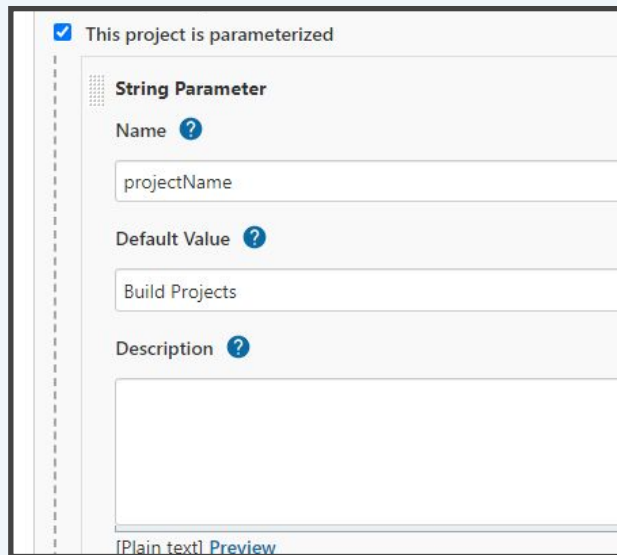
Our group decided to use Bitbucket as a repository to store all of the scripts used to start and stop the containers. This would ensure that these instructions weren't lost in case Jenkins was reset or lost its data.

Using a git repository in Jenkins also provides a method of version control as git will always keep copies of our previous builds.



```
hackathon3 / startpet
1  #!/bin/bash
2
3  cd "/var/lib/jenkins/workspace/${projectName}"
4  cd containers
5  docker-compose up -d petclinic
6
```

The first step was adding the parameter “projectName” to replace the original name which was “Build Projects”. This ensures that the process name can easily be changed in the future.



☒ This project is parameterized

**String Parameter**

Name [?](#)

projectName

Default Value [?](#)

Build Projects

Description [?](#)

[Plain text] [Preview](#)



# Source code management

This is an example of how we got Jenkins to read from our git repository.

The git repository link is entered in Jenkins

The Jenkins shell only has the titles of the bitbucket files rather than the source code

Additionally, some Jenkins projects (such as the one pictured) were made to run multiple scripts at once to reduce manual touch.

Source Code Management

☐ None  
☒ Git

Repositories

Repository URL  
https://bitbucket.org/katrina057/hackathon3.git

Credentials  
- none - [Add](#)

Advanced...  
Add Repository

Branches to build

Branch Specifier (blank for 'any')  
\*/main

Build

Execute shell

Command  
bash ./startddb  
bash ./startpet  
bash ./startwordpress

[See the list of available environment variables](#)

Advanced...

# Release

## Roll Back

### Source Code Management

☐ None

☒ Git

#### Repositories

##### Repository URL

##### Credentials

#### Branches to build

##### Branch Specifier (blank for 'any')

#### Branches to build

##### Branch Specifier (blank for 'any')

# ITRS

A **real-time application performance monitoring tool** that helps you manage the increasingly complex interdependencies between your infrastructure, applications and business services.

# ITRS Knowledge Base

We added a knowledge base to the percent utilisation for the CPU. This is useful for when the value hits the warning threshold as when you click the knowledge base it links you to a website explaining what to do if a linux machine's CPU is high. The knowledge base is denoted by an "i" on the cells which include one.

Name

Fix High CPU

URL template elements

options

literal

https://www.maketecheasier.com/fix-high-cpu-usage-lin

[Add new](#)

KBA set - High CPU

Normal

Name

High CPU

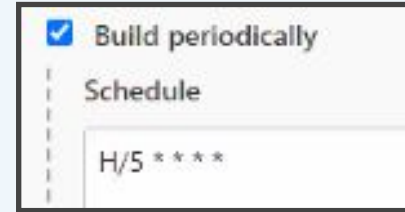
KBA

Label	URL elements	Targets
<div><div></div></div>	<div>URL element</div> <div>URL elements...</div> <div>URL element</div>	<div>Target</div> <div>Target: ...ks"]]/dataview[(@name="CPU")]/rows/row/c</div>

percentUtilisation
<div><div></div> 10.10 %</div>
<div><div></div> 10.14 %</div>
<div><div></div> 8.54 %</div>

# Petclinic/Wordpress application status monitoring and automation

Originally there was a build per application in Jenkins which was automated to run every 5 minutes to check if the site was still up.

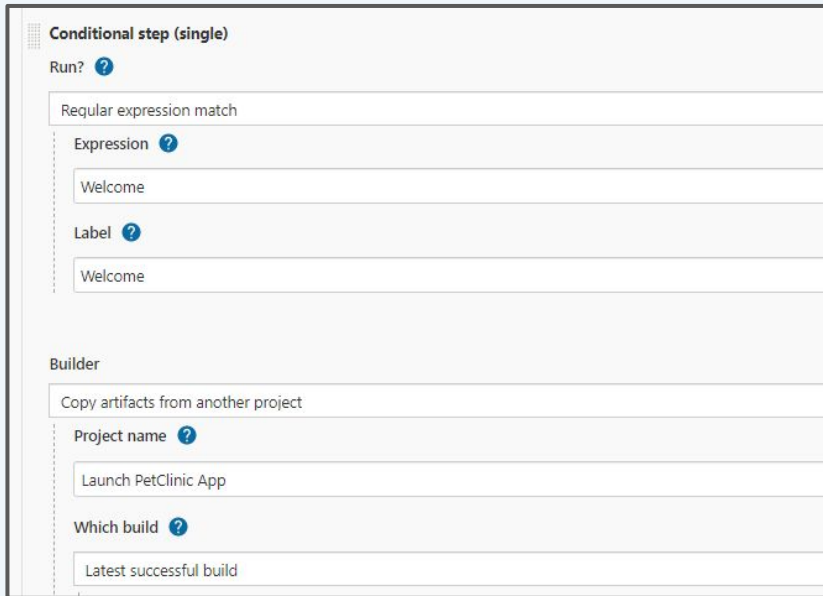


The way the status was checked on both Petclinic and WordPress was by searching for the word “Welcome”

```
Execute shell
Command
if curl 'http://ps2.eu.conygre.com:1080' | grep 'Welcome' ; then
    echo "Petclinic is up"
else
    echo "Petclinic is| down"
fi
```

# Petclinic/Wordpress application status monitoring and automation

If the first shell script could not find the word “Welcome” it relaunched the latest stable build of the application



The image shows a Jenkins configuration interface for a 'Conditional step (single)'. It is divided into two main sections: 'Run?' and 'Builder'.

**Run? Section:**

- Regular expression match:** This section contains three input fields:
  - Expression:** Contains the text 'Welcome'.
  - Label:** Contains the text 'Welcome'.

**Builder Section:**

- Copy artifacts from another project:** This section contains three input fields:
  - Project name:** Contains the text 'Launch PetClinic App'.
  - Which build:** Contains the text 'Latest successful build'.

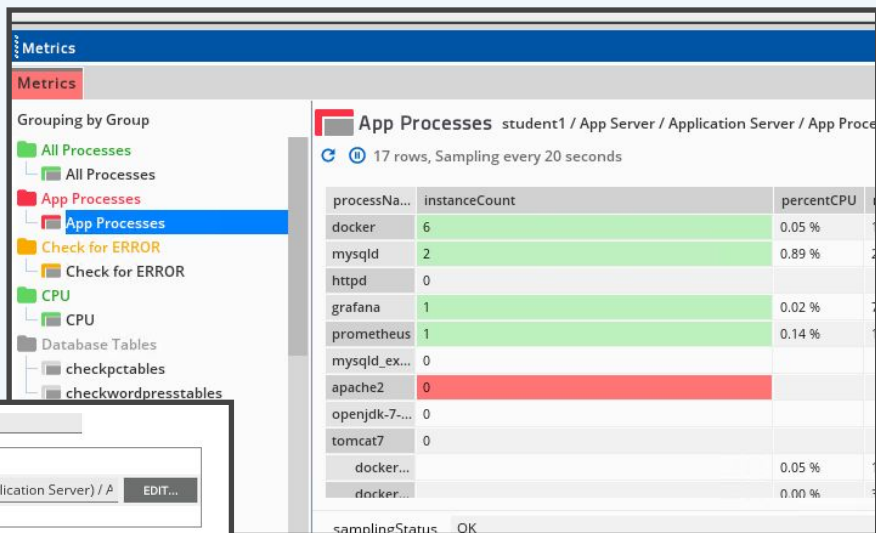
The issue with using Jenkins was that building a project every 5 minutes to check the status of the site meant that it checked less frequently than ITRS could (it could check more frequently but this would use a very high level of resources to run), and there was also no way to delay samples. The responsibility for checking if the sites were up was moved to ITRS, however, the launch-related parts of the projects were kept on Jenkins for the advantage of git source code management.

A disadvantage of moving the automation to ITRS is that Jenkins has the option to launch only the last stable build. ITRS will attempt to launch the latest build. A rollback plan would need to be implemented in future builds when using ITRS.

# Petclinic/Wordpress application status monitoring and automation

ITRS now checks the applications are up every 20 seconds using the 'instanceCount' feature

A rule was set so that if the instances become 0 then the severity would be marked critical (highlighted red in the previous screenshot) and ITRS would run an action to restart the application



Managed entities

- Dynamic entities
- Types
- Samplers
- Sampler includes
- Actions
- Effects
- Annotations
- Commands
- Scheduled commands
- Rules

Petclinic down

Name: Petclinic down

Targets: Target

/ student1 / App Server / Application Server / App Processes(type=Application Server) / A

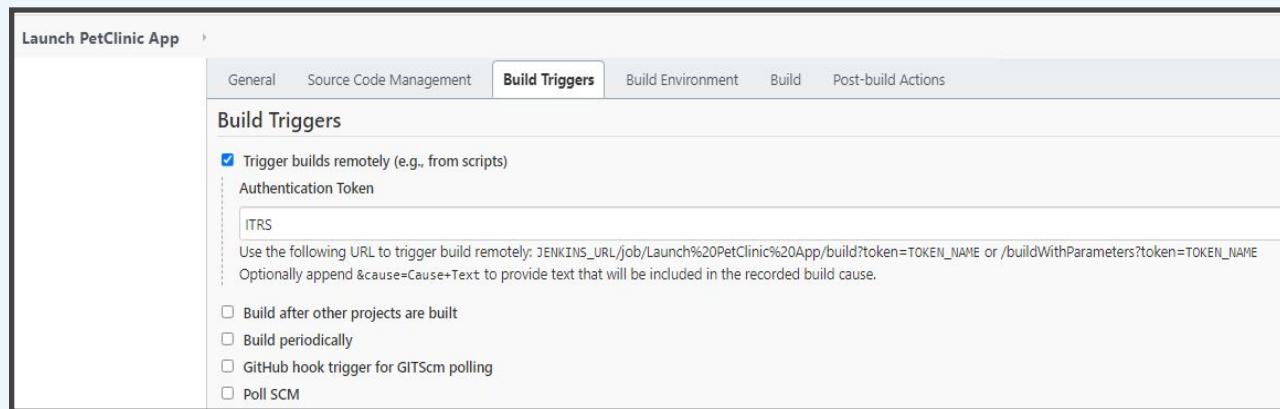
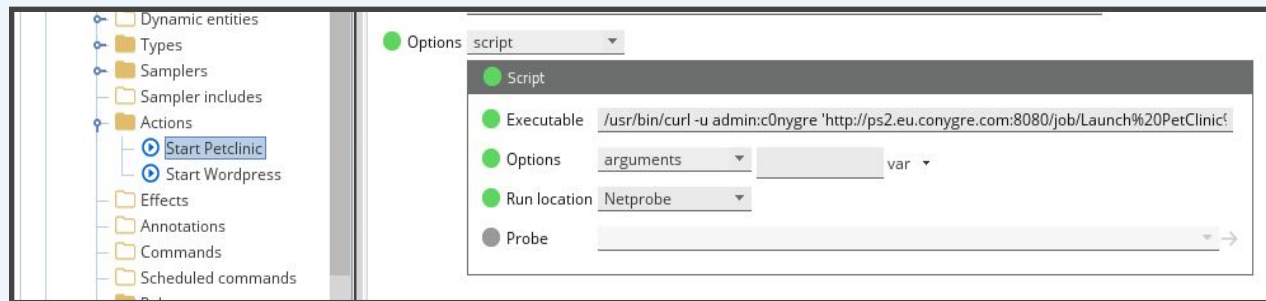
Priority: 1

Block:

```
1 if value = 0 then
2   delay 2 samples
3   severity critical
4   run "Start Petclinic"
5 else
6   severity ok
7 endif
8
9
```

# Petclinic/Wordpress application status monitoring and automation

The action ran by the previous rule links to a Jenkins build to launch the relevant application





# Resiliency

All of the Jenkins jobs created to build, start and stop the containers were stored on a Bitbucket repository and referenced via parameters

The image shows a Jenkins job configuration page for a job named 'hackathon3 / startpet'. The configuration is divided into several sections:

- Script:** A shell script is shown in a text editor:

```
1 #!/bin/bash
2
3 cd "/var/lib/jenkins/workspace/${projectName}"
4 cd containers
5 docker-compose up -d petclinic
6
7 echo "Petclinic Container version is ${BUILD_NUMBER}"
```
- Source Code Management:** This section is configured for Git. The Repository URL is 'https://bitbucket.org/katrina057/hackathon3.git'. The Credentials dropdown is set to 'none'. The Branches to build section is set to '\*/main'.
- Build:** This section is configured to 'Execute shell'. The Command field contains:

```
bash ./startdb
bash ./startpet
bash ./startwordpress
```
- Parameter:** A 'String Parameter' is defined with the name 'projectName' and a default value of 'Build Projects'.

At the bottom left, there is a link to '[Plain text] Preview'.

# Backing up application databases

Backing up the database is done via a scheduled Jenkins project. The schedules were based around potential business needs, for example, PetClinic is set to back up every 2 hours 30 minutes past the hour between 9:30-7:30 as it is designed to be used internally by a vet clinic.

The code for the backup would look for a current backup directory and make one if it doesn't already exist. It would then take the copy of the relevant sql database and put it into the backup directory.

### Build Triggers

☐ Trigger builds remotely (e.g., from scripts)

☐ Build after other projects are built

☒ Build periodically

Schedule

30 H(9-19)/2 \* \* \*

Would last have run at Thursday, October 21, 2021 7:30:00 PM UTC; wo

### Build

Execute shell

Command

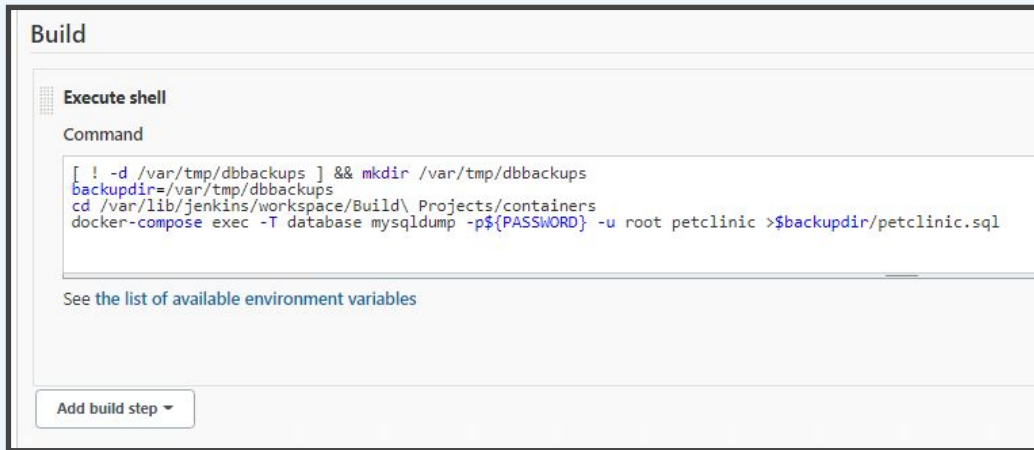
```
[ ! -d /var/tmp/dbbackups ] && mkdir /var/tmp/dbbackups
backupdir=/var/tmp/dbbackups
cd /var/lib/jenkins/workspace/Build\ Projects/containers
docker-compose exec -T database mysqldump -p${PASSWORD} -u root petclinic >${backupdir}/petclinic.sql
```

[See the list of available environment variables](#)

Add build step ▾

## Backing up application databases

The code for the backup would look for a current backup directory and make one if it doesn't already exist. It would then take the copy of the relevant sql database and put it into the backup directory.



Backing up of the database is still done entirely through Jenkins for both applications. In a future build it would be more ideal to move this code over to git in case the Jenkins job is ever lost.

# Monitoring application databases

Backups would not be as useful if there is not a way to restore the databases from the created backup directories.

Originally, an automation was set up in Jenkins which would search for a word in the body of one of the tables inside the database on a set schedule. If it could not find the word then it would run an additional shell script (from the bitbucket repository) to restore the database from the backup. Displayed is the Jenkins automation to both check and restore the backup. It ran every 5 minutes on a schedule.

Build

Execute shell

Command

```
cd /var/lib/jenkins/workspace/Build\ Projects\containers  
docker-compose exec -T database mysql -uroot -ppetclinic wordpress -e 'SELECT "admin" FROM wp_users;'
```

[See the list of available environment variables](#)

Advanced...

Conditional step (single)

Run?

Not

!

Regular expression match

Expression

admin

Label

admin

Advanced...

Builder

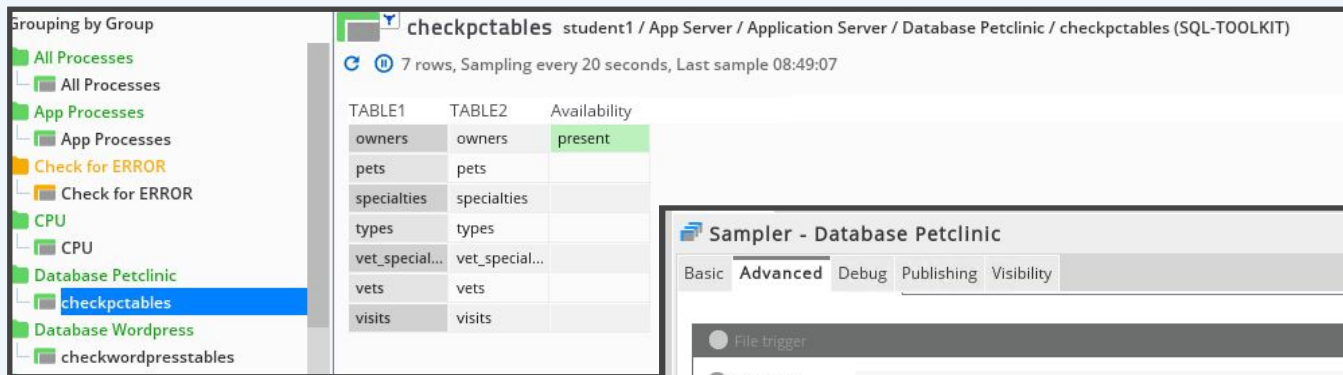
Execute shell

Command

```
bash ./RestoreWordpress
```

# Monitoring application databases

The Jenkins build/automation wasn't practical as the build running every 5 minutes would use a lot of resources; there was also no easy way to snooze the automation from running if you needed to take the database down for a period of time. For this reason the first part of the automation (checking the database still exists) was moved to ITRS.



Grouping by Group

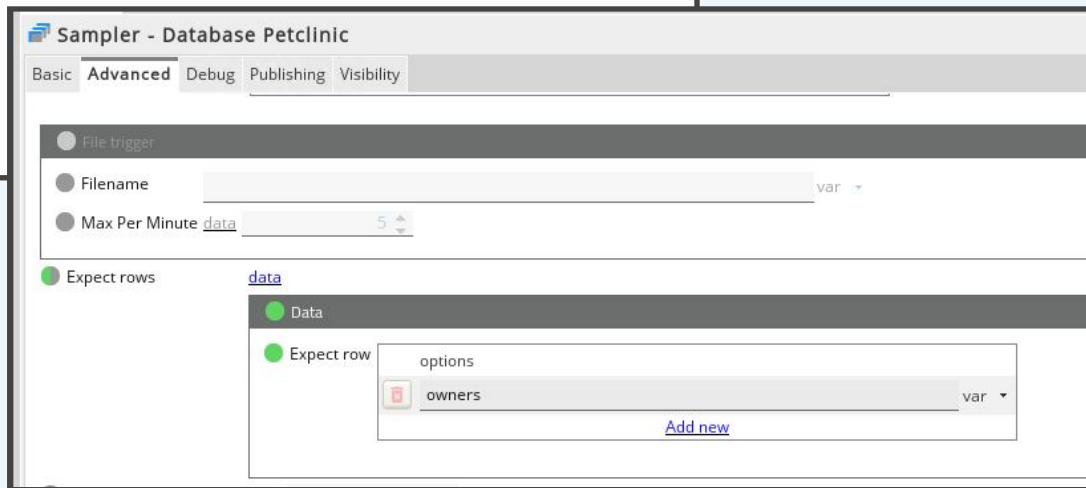
- All Processes
  - All Processes
- App Processes
  - App Processes
- Check for ERROR
- CPU
  - CPU
- Database Petclinic
  - checkpctables**
  - Database Wordpress
    - checkwordpressables

checkpctables student1 / App Server / Application Server / Database Petclinic / checkpctables (SQL-TOOLKIT)

7 rows, Sampling every 20 seconds, Last sample 08:49:07

TABLE1	TABLE2	Availability
owners	owners	present
pets	pets	
specialties	specialties	
types	types	
vet_special...	vet_special...	
vets	vets	
visits	visits	

In the 'advanced' tab a cell of the table could be marked as 'present' or 'absent' by telling ITRS to expect it. This was a useful tool for monitoring if the database is present.



**Sampler - Database Petclinic**

Basic **Advanced** Debug Publishing Visibility

☐ File trigger

☐ Filename

☐ Max Per Minute data

☒ Expect rows data

☒ Data

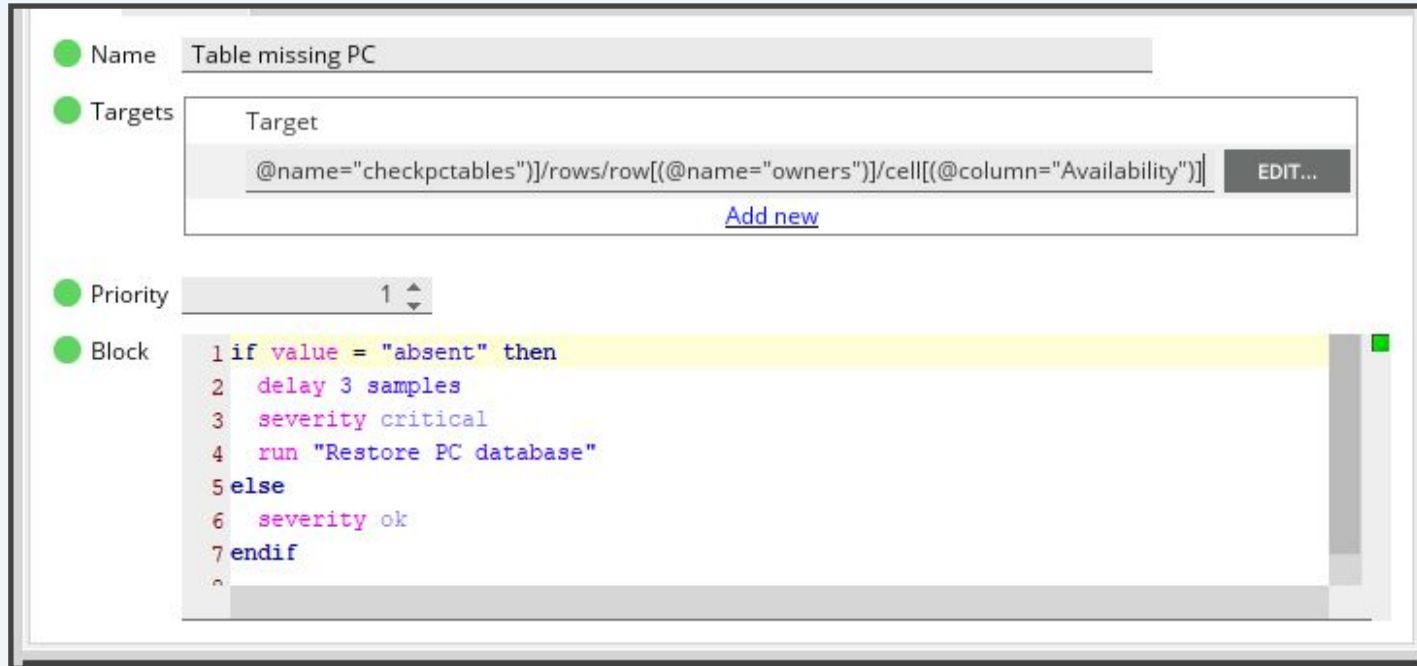
☒ Expect row

options

[Add new](#)

# Monitoring and automating application databases

The rule set checks to see if the value of 'owners' is present in the table. If it is not then it delays 3 samples (checks 3 times over 20 seconds each check) then will mark the severity as critical (turns the cell red) and run the action to restore the PetClinic database.



The screenshot shows a configuration window for a monitoring rule. It has four main sections: Name, Targets, Priority, and Block. The 'Name' field is 'Table missing PC'. The 'Targets' section shows a single target with a complex XPath-like expression. The 'Priority' is set to 1. The 'Block' section contains a script that checks for the presence of 'owners' in a table and either delays and marks as critical or marks as ok.

**Name** Table missing PC

**Targets**

Target
@name="checkpctables"))/rows/row[(@name="owners")]/cell[(@column="Availability")]

[Add new](#) EDIT...

**Priority** 1

**Block**

```
1 if value = "absent" then
2   delay 3 samples
3   severity critical
4   run "Restore PC database"
5 else
6   severity ok
7 endif
```

# Monitoring and automating application databases

The action 'Restore PC Database' links back to the edited original Jenkins job which runs from a git repository.

## Build Triggers

☒ Trigger builds remotely (e.g., from scripts)

Authentication Token

ITRS

Use the following URL to trigger build remotely: JENKINS\_URL/job/Restore%20Database%20petclinic/build?token=TOKEN\_NAME or /buildWithParameters?token=TOKEN\_NAME

Optionally append &cause=Cause+Text to provide text that will be included in the recorded build cause.

- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ GitHub hook trigger for GITScm polling
- ☐ Poll SCM

## Build Environment

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s)
- ☐ Provide Configuration files
- ☐ Send files or execute commands over SSH before the build starts
- ☐ Send files or execute commands over SSH after the build runs
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output
- ☐ Execute shell script on remote host using ssh
- ☐ SSH Agent

## Build

Execute shell

Command

bash ./RestorePetclinic

Name Restore PC database

Options script

Script

Executable /usr/bin/curl -u admin:c0nygre 'http://ps2.eu.conygre.com:8080/job/Restore%20Databas

Options arguments var

Run location Netprobe

Probe

hackathon3 / RestorePetclinic

```
1 cd "/var/lib/jenkins/workspace/${projectName}"
2 cd containers
3
4 docker-compose exec -T database mysql -uroot -ppetclinic -e "create database petclinic;"
5 docker-compose exec -T database mysql -uroot -ppetclinic petclinic </var/tmp/dbbackups/petclinic.sql
```

# Database status monitoring and automation

A rule was set so that if the instances become 0 then the severity would be marked critical and ITRS would run an action to restart the application.

The screenshot displays the ITRS configuration interface for a rule named "MySQL Database down". On the left, a tree view shows the hierarchy: Dynamic entities > Types > Samplers > Sampler includes > Actions > Effects > Annotations > Commands > Scheduled commands > Rules. The "Rules" folder is expanded, showing several rules, with "MySQL Database down" selected. The main configuration area on the right shows the following details:

- Name:** MySQL Database down
- Targets:** A text field containing the path "/ student1 / App Server / Application Server / App Processes(type=Application Server) / App Processes / mysql". An "EDIT..." button is located to the right of the text field. Below the text field is a link labeled "Add new".
- Priority:** 1
- Block:** A code editor containing the following logic:

```
1 if value = 0 then
2   delay 2 samples
3   severity critical
4   run "Start Database"
5 else
6   severity ok
7 endif
8
9
10
```



# Database status monitoring and automation

The action ran by the previous rule links to a Jenkins build to launch the relevant application

**Action - Start Database**

**Basic** Advanced

Name

Start Database

Options

script

Script

Executable

/usr/bin/curl -u admin:c0nygre 'http://ps2.eu.conygre.com:8080/job/Launch%20Database'

Options

arguments

var

Run location

Netprobe

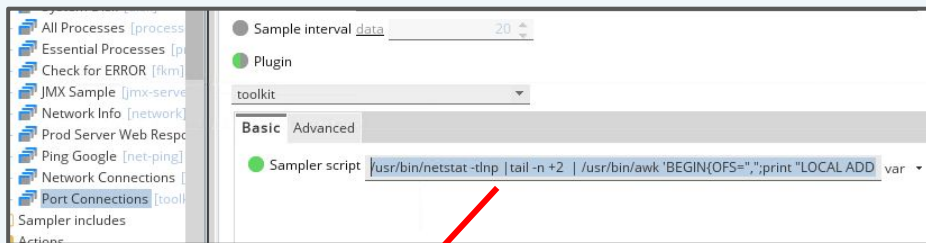
Probe

## Monitoring and automating application databases, future build recommendations

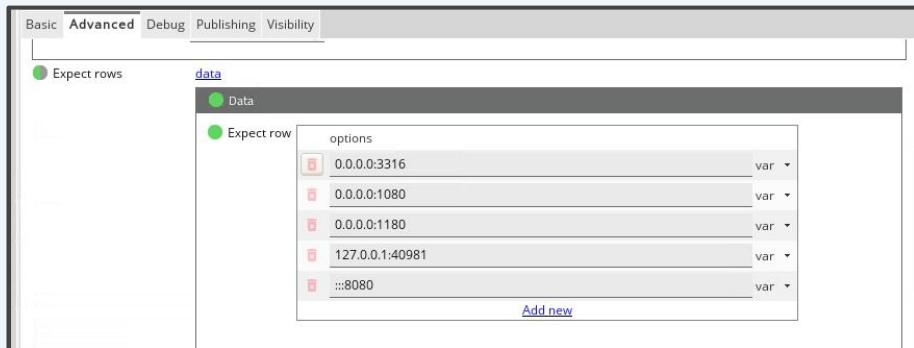
- In a future build the automation for restoring the database could be improved by alerting when it has had to restore and/or including another automation which prevents it from attempting to restore too many times in a row. There is a risk with automating the database to restore from backup that it could restore from a corrupted backup or get stuck in a loop of always trying to restore without enough warning that this is happening.
- A way to keep several backups on the system would also be useful in this case as currently every time the database backs up it overwrites the previous backup. A better roll back plan is needed for if an incident happens and the quality of the backup has been compromised.

# Monitoring port connections

LOCAL ADDRESS	PID/PROGRAM	Availa... ▼ 1
127.0.0.1:40981	3174/containerd	present
0.0.0.0:1080	4407/docker-proxy	present
0.0.0.0:1180	4462/docker-proxy	present
:::8080	4150/java	present
0.0.0.0:3316		absent



`/usr/bin/netstat -tlnp | tail -n +2 | /usr/bin/awk 'BEGIN{OFS=",";print "LOCAL ADDRESS,PID/PROGRAM"}{print $4,$NF}'`



A new type was added to monitor if there was anything on the expected ports. This means that if one of the containers goes down the local address would be marked as critical.

This was made with the idea that it could be built upon in a future build where the name connected to the port could be monitored to make sure the applications are running on the expected ports.

# Capacity

For considerations of increased capacity for the application, we would look at the following:

- Consider container storage ecosystem (CaaS, PaaS etc)
  - CaaS (Container as a Service) is the most common allowing reduction in operating expenses, scaling up or down as needed and developers can respond quickly deploying new or updated software
- Changing to Openshift over using Docker Compose
  - Allows for easy deployment and scaling of containers
  - Adds management tools for containers
  - Abstracted cluster orchestration (used for control and automating containers life cycle)
- Increasing resources
  - Increasing the resources available if it becomes a bottleneck for the application functioning
  - Use of load balancers to help with distribution of application traffic across many servers - increase capacity and reliability of the applications

## Cyber (Future Improvements)

Runtime Hardening - securing the container with rules and giving it the lowest level of privileges/access possible for it to be able to adequately carry out its job.

1. SELinux
2. Capabilities
3. File Systems
4. Seccomp

Image Hardening (also known as Image Stripping) - getting rid of any unnecessary files and services within the image that aren't being utilised.

# Questions

**Thank you for listening!**