



SECURE VM WITH AZURITE BLOB STORAGE BACKUP

A step-by-step journey of securing a Linux server
and automating backups

Summary

This project shows how I built a secure Linux server on VirtualBox and set up automated backups to Azurite (an Azure Storage emulator). The process included hardening the VM with a firewall and Fail2Ban, configuring SSH for safe access, and using scripts with azcopy to send backups into blob storage. Microsoft Storage Explorer was used to confirm everything worked. Finally, I automated the process with cron and added a dedicated backup user for better security.



Tebogo Matseding
tmatseding@outlook.com

Table of Contents

Introduction	Restart sshd
First Log on	Restart ssh
Update	Verify changes didnt break anything
UFW status	Generate Keys
Close all ports	Transfer Public Key
Allow SSH traffic	Permissions
Enable UFW	Append Contents
Verify rules were applied	Try logging with root
Install OpenSSH-Server	Log In as user
OpenSSH-Server Status	Install Fail2ban
Config rules	Making copy
Permit Root Login	Email Setting
Public Key Authentication	SSHD rules
Password Authenitication &	Recidive
Permit Empty Password	Fail2ban Jails
PAM	

Test Fail2ban	Script worked
Logging using PuTTY	Verify Upload
Create Script	Create Container
Script permissions	Official Backup Script
Script Complete	Backup is successful
Verify Creation	Verify Backup
Download Microsoft Azure Storage Explorer	Create Backupuser
Download Nodejs	Move script and hand over ownership
Install Azurite	Update Script
Download azcopy	Verify Backup User can use script
Unzip and move azcopy	Verify Backup
Create A Firewall Rule	Crontab and schedule backup
Create Test Container	Test Crontab
Get SAS Token	Crontab worked
Verify IP Address	Done
Test script for backup	

START

Introduction

This guide walks you through creating a secure Linux virtual machine, setting up automated backups using Azurite (an Azure Storage emulator), and ensuring everything is protected with firewall rules, SSH configuration, and Fail2Ban.

First Log on

After starting your VM in VirtualBox, log in using the default credentials. You will see a terminal prompt where all commands will be entered.

```

Ubuntu 25.04 awrite-srv.txt
awrite-srv login: teboqo
password:
Welcome to Ubuntu 25.04 (GNU/Linux 6.14.0-29-generic wsl64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/support

System information as of Fri Aug 22 07:42:08 PM UTC 2025

System load: 0.39
Usage of /: 68.0% of 16.0Gib
Memory usage: 68
Swap usage: 0
Processes: 141
Users logged in: 0
IPv4 address for enp0s3: 10.0.2.15
IPv6 address for enp0s3: fd17:625c:f037:2::a4a:27f1:feec:b6
```

Update

(using `sudo apt update` & `sudo apt upgrade`)

Make sure the system is up-to-date with the latest software and security patches

```

root@kali:~# curl -s https://api.github.com/repos/SecWiki/SecWiki
{"name":"SecWiki","owner":{"login":"SecWiki","id":10497007,"type":"User"},
"private":false,"html_url":"https://github.com/SecWiki/SecWiki","description":"\u5b66\u4e60\u533a\u573a",
"fork":false,"url":"https://api.github.com/repos/SecWiki/SecWiki",
"forks_url":"https://api.github.com/repos/SecWiki/SecWiki/forks",
"keys_url":"https://api.github.com/repos/SecWiki/SecWiki/keys",
"collaborators_url":"https://api.github.com/repos/SecWiki/SecWiki/collaborators",
"teams_url":"https://api.github.com/repos/SecWiki/SecWiki/teams",
"branches_url":"https://api.github.com/repos/SecWiki/SecWiki/branches",
"tags_url":"https://api.github.com/repos/SecWiki/SecWiki/tags",
"commits_url":"https://api.github.com/repos/SecWiki/SecWiki/commits",
"statuses_url":"https://api.github.com/repos/SecWiki/SecWiki/statuses/{sha}",
"issue_url":"https://api.github.com/repos/SecWiki/SecWiki/issues",
"contents_url":"https://api.github.com/repos/SecWiki/SecWiki/contents/{+}",
"compare_url":"https://api.github.com/repos/SecWiki/SecWiki/compare/{base}...{head}",
"pulls_url":"https://api.github.com/repos/SecWiki/SecWiki/pulls",
"stargazers_url":"https://api.github.com/repos/SecWiki/SecWiki/stargazers",
"contributors_url":"https://api.github.com/repos/SecWiki/SecWiki/contributors",
"subscribers_url":"https://api.github.com/repos/SecWiki/SecWiki/subscribers",
"subscription_url":"https://api.github.com/repos/SecWiki/SecWiki/subscription",
"archive_url":"https://api.github.com/repos/SecWiki/SecWiki/{archive_format}/{ref}.zip",
"zipball_url":"https://api.github.com/repos/SecWiki/SecWiki/zipball/{ref}",
"tarball_url":"https://api.github.com/repos/SecWiki/SecWiki/tarball/{ref}",
"language":"python",
"has_issues":true,"has_projects":true,"has_downloads":true,"has_wiki":true,"has_pages":true,
"mirror_url":null,"license":{"key":"gpl-3.0","name":"GNU General Public License v3.0",
"spdx_id":"GPL-3.0","url":"https://www.gnu.org/licenses/gpl-3.0.html","node_id":"MDDEI5"},
"updated_at":"2020-07-20T06:00:00Z","ssh_url":"git@github.com:SecWiki/SecWiki.git",
"git_url":"git://github.com:SecWiki/SecWiki.git","git_ref_url":"https://api.github.com/repos/SecWiki/SecWiki/git/ref/{ref}"}

```

UFW status

(using `sudo systemctl status ufw`)

Check the status of the firewall (UFW – Uncomplicated Firewall)
If inactive, we'll enable it later after setting rules.

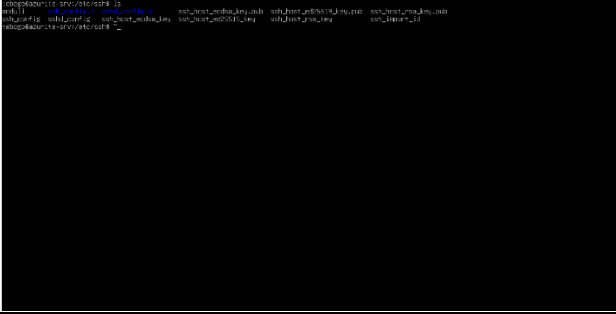
[illegible]

Close all ports

(using sudo ufw default deny incoming)

Start by blocking all incoming connections to make the server secure

```
telnet@zerate:~$ sudo ufw default deny incoming
default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
telnet@zerate:~$
```


<h2>Config rules</h2> <p>(using sudo nano /etc/ssh/sshd_config)</p> <p>Edit the SSH configuration file to harden access</p>	
<h2>Permit Root Login</h2> <p>(yes)</p> <p>For security, root login should be disabled</p>	<pre>#HostKey /etc/ssh/ssh_host_rsa_key #HostKey /etc/ssh/ssh_host_ecdsa_key #HostKey /etc/ssh/ssh_host_ed25519_key # Ciphers and keying #RekeyLimit default none # Logging #SyslogFacility AUTH #LogLevel INFO # Authentication: #LoginGraceTime 2m PermitRootLogin yes #StrictModes yes #MaxAuthTries 6 #MaxSessions 10</pre>
<h2>Public Key Authentication</h2> <p>(yes)</p> <p>Enable key-based login for secure access</p>	<pre># Ciphers and keying #RekeyLimit default none # Logging #SyslogFacility AUTH #LogLevel INFO # Authentication: #LoginGraceTime 2m PermitRootLogin yes #StrictModes yes #MaxAuthTries 6 #MaxSessions 10 PubkeyAuthentication yes</pre>
<h2>Password Authentication & Permit Empty Password</h2> <p>(no & no)</p> <p>Disable password logins and empty passwords</p>	<pre>#AuthorizedKeysCommand none #AuthorizedKeysCommandUser nobody # For this to work you will also need host keys in /etc/ssh/ssh_known_hosts #HostbasedAuthentication no # Change to yes if you don't trust ~/.ssh/known_hosts for # HostbasedAuthentication #IgnoreUserKnownHosts no # Don't read the user's ~/.rhosts and ~/.shosts files #IgnoreRhosts yes # To disable tunneled clear text passwords, change to no here! PasswordAuthentication no PermitEmptyPasswords no</pre>

Restart sshd

(using `sudo systemctl restart sshd`)

Apply configuration changes by restarting SSH

```
root@azurite:~# sudo systemctl restart sshd
```

Restart ssh

(using `sudo service ssh restart`)

Optionally, restart the SSH service fully

```
root@azurite:~# sudo service ssh restart
```

Verify changes didnt break anything

```
root@azurite:~# sudo systemctl restart sshd
root@azurite:~# sudo systemctl status ssh
ssh.service - OpenSSH Secure Shell server
Loaded: loaded (/usr/lib/systemd/system/ssh.service; disabled; preset: enabled)
Active: active (running) since Sat 2023-08-23 06:55:12 UTC; 25s ago
Process: 6954105276413f8ed3c27527d6fa
TriggeredBy: sshd.service
Docs: man:sshd(8)
       man:sshd_config(5)
Process: 2509 /usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
Main PID: 2572 (sshd)
Tasks: 1 (limit: 799)
Memory: 1.2M (peak: 1.7M)
CGroup: /system.slice/ssh.service
        └─2572 "sshd: user@ubuntu:0.0 (listener) * of 10-100 startups"

Aug 23 06:55:12 azurite-01 sshd[2572]: Starting sshd service - OpenSSH Secure Shell server...
Aug 23 06:55:12 azurite-01 sshd[2572]: Server listening on 0.0.0.0 port 22.
Aug 23 06:55:12 azurite-01 sshd[2572]: Server listening on :: port 22.
Aug 23 06:55:12 azurite-01 systemd[1]: Started ssh.service - OpenSSH Secure Shell server.
root@azurite:~#
```

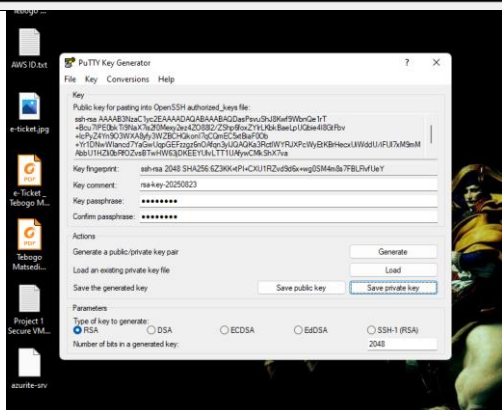
Generate Keys

(using PuTTY)

Use PuTTYgen on your Windows machine to generate an SSH key pair.

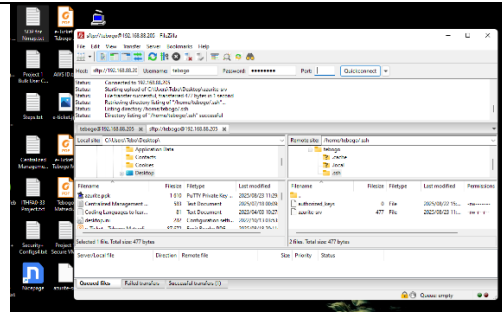
Save the private key (.ppk) on your local computer.

Save the public key (.pub) file, which will be uploaded to the server.

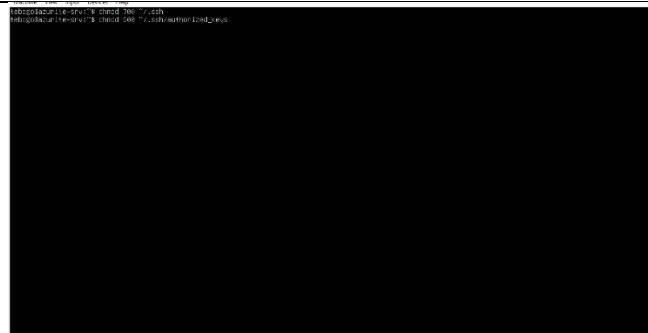


Open FileZilla and connect to your VM using the temporary username/password.

Upload the public key file



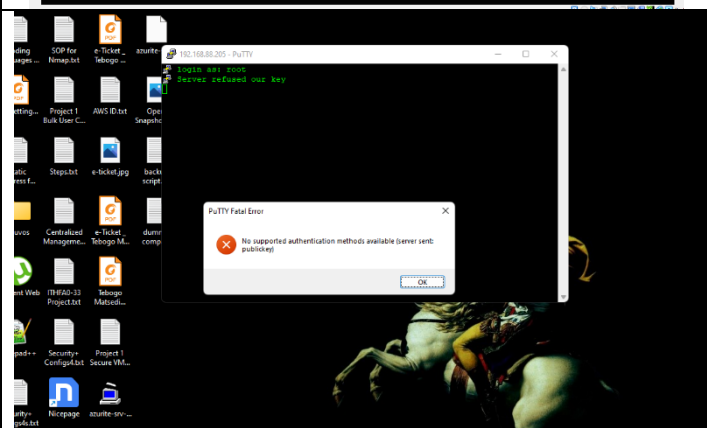
Ensure .ssh folder and files have correct permissions



Append the public key to
authorized_keys



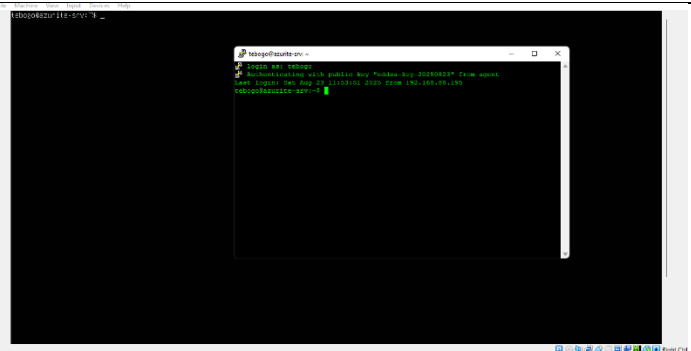
Confirm root login is blocked;
only your user can log in.



Log In as user

(using my private key)

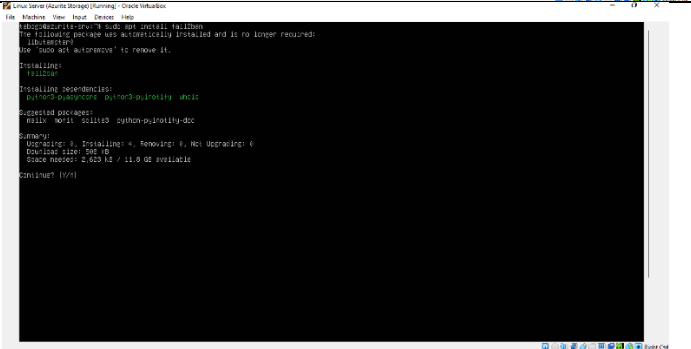
Test SSH login with the normal user account.



Install Fail2ban

(using sudo apt install fail2ban)

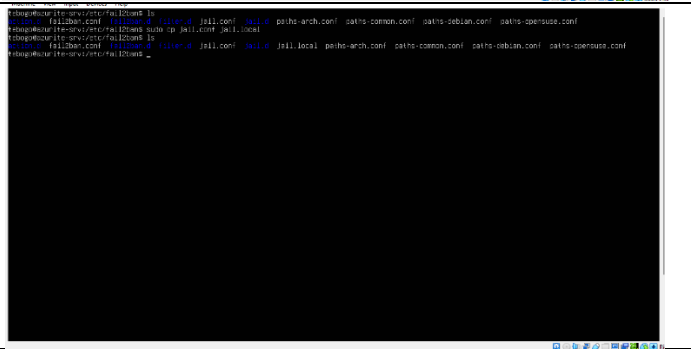
Install Fail2Ban to block repeated login attempts



Making copy

(of the jail.conf to jail.local)

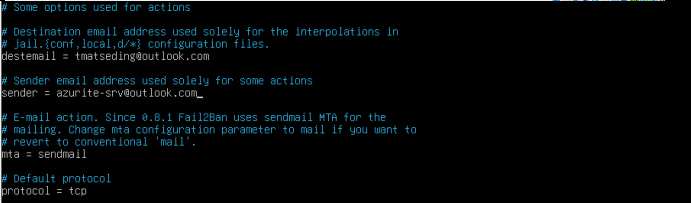
Backup the default Fail2Ban configuration



Email Setting

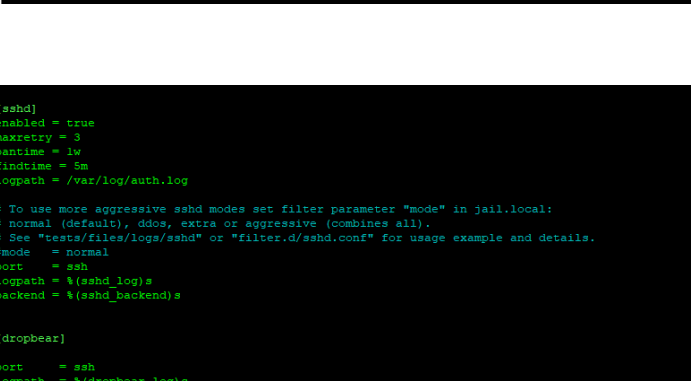
(receive alerts if someone is trying to access the server)

Set email notifications in jail.local to get alerts for blocked attempts.



SSHD rules

Enable SSH protection in Fail2Ban config.



Enable recidive jail to block repeated offenders for longer periods.

⌘G Help ⌘O Write Out ⌘F Where Is ⌘K Cut ⌘T Execute ⌘C Locati

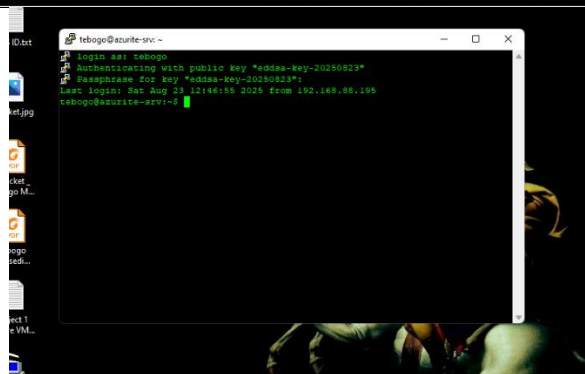
Check active jails to confirm SSH protection

```
root@kali:~# sudo fail2ban-client status
Status
- Number of jail:      1
- Fail list:           none
root@kali:~#
```

(by attempting to log in with
another computer)

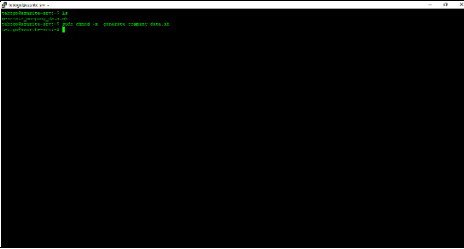
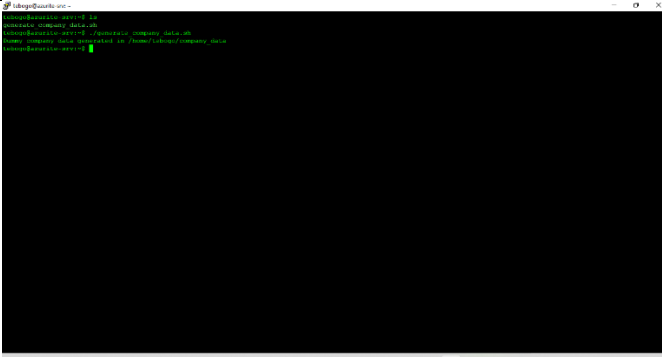

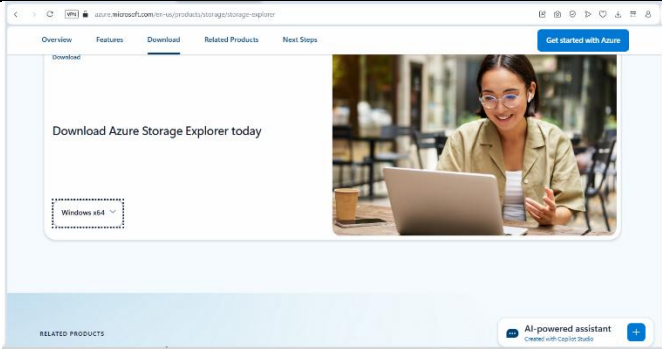

[illegible]

Monitor SSH login attempts using PuTTY or local logs

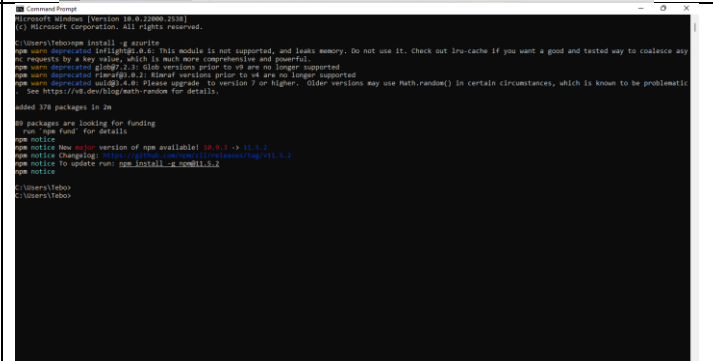


(that generates a dummy
company information)

[illegible]

<p>Script permissions</p>	
<p>Script Complete</p>	
<p>Verify Creation</p>	
<p>Download Microsoft Azure Storage Explorer</p>	
<p>Download Nodejs</p>	

Install Azurite

[illegible]

Download azcopy

[illegible]

Unzip and move

azcopy

Extract and move it to a folder in your
PATH for easy use

azcopy

Extract and move it to a folder in your
PATH for easy use

azcopy

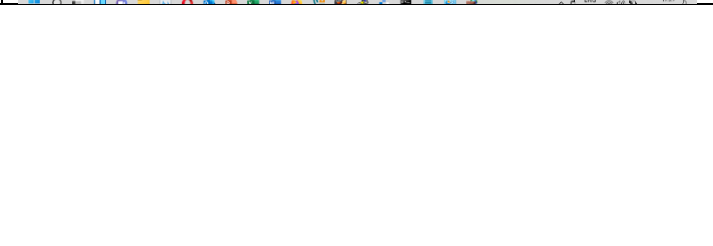
Extract and move it to a folder in your
PATH for easy use

Create A Firewall Rule

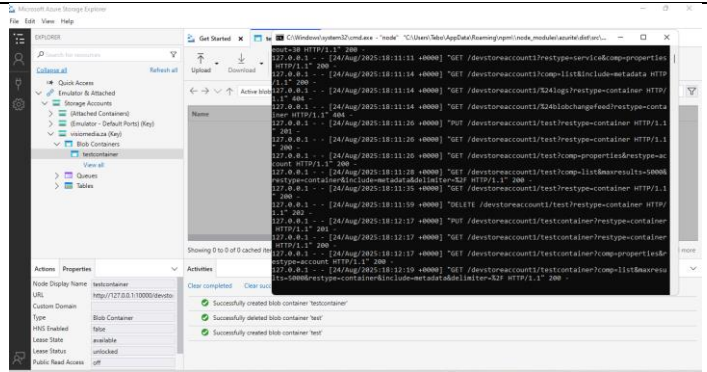
Ensure Azurite can be accessed safely through firewall if needed.

Create A Firewall Rule

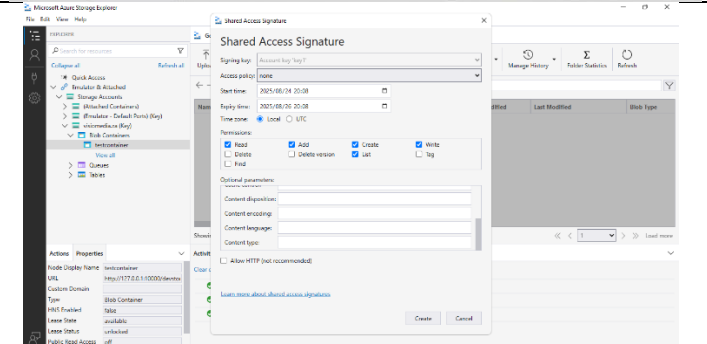
Ensure Azurite can be accessed safely through firewall if needed.



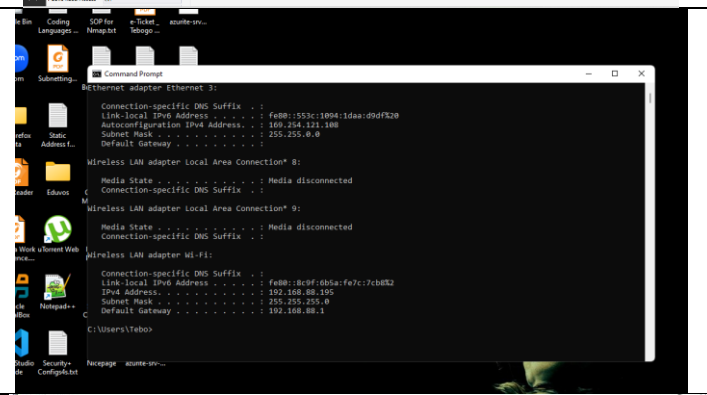
Create Test Container
Create a test storage container in Azurite for initial uploads.



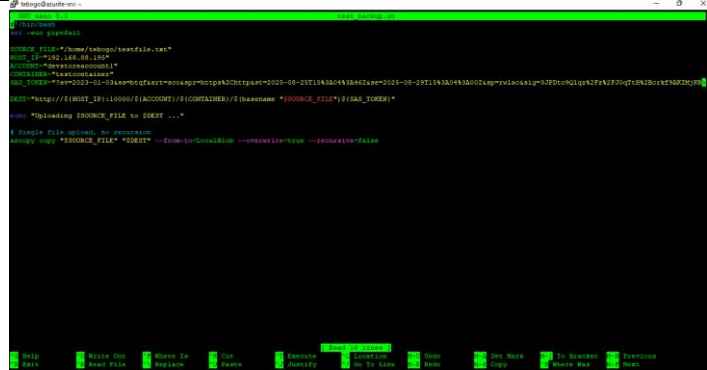
Get SAS Token
Generate a SAS token to authenticate uploads.



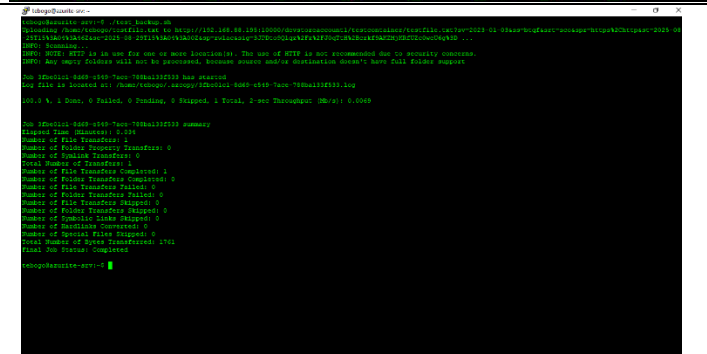
Verify IP Address



Test script for backup
Run the backup script and check it completes without errors

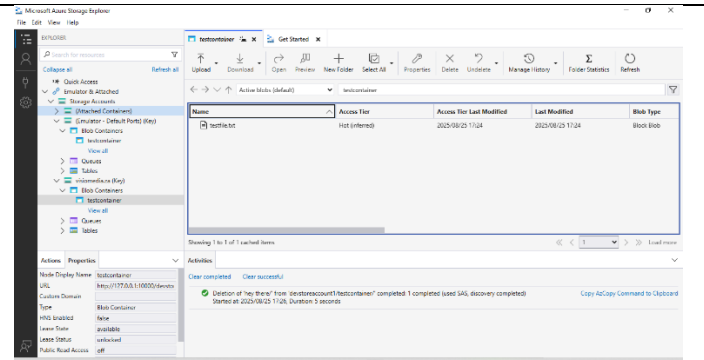


Script worked



Verify Upload

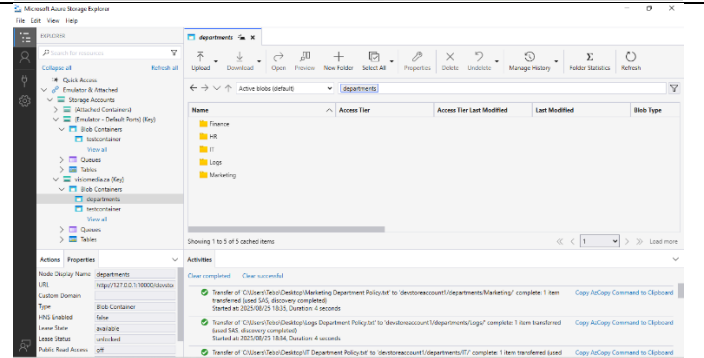
Use Storage Explorer to check that files appear in the container



Create Container

(with corresponding folders)

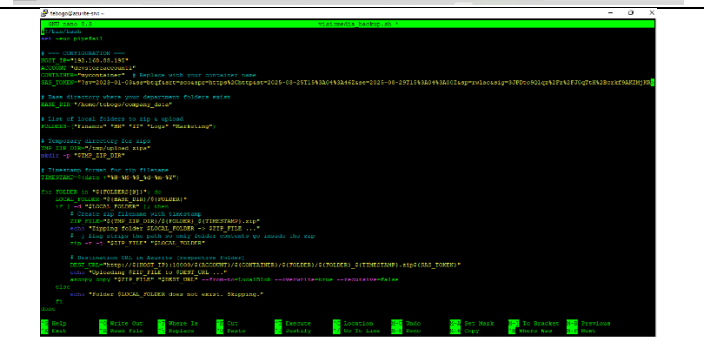
Create a dedicated container for official backups



Official Backup Script

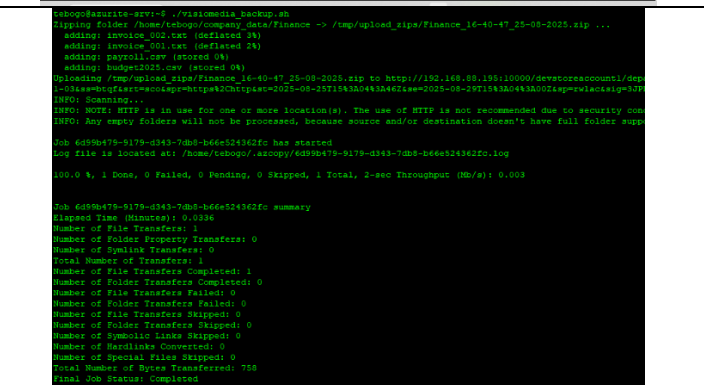
Update the script to use the official container and SAS token

[backup script](#)

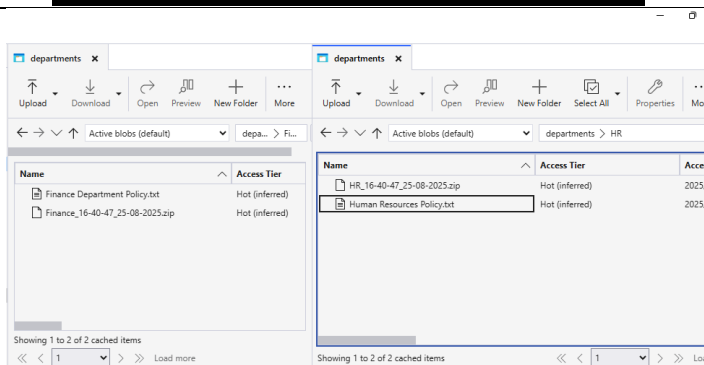


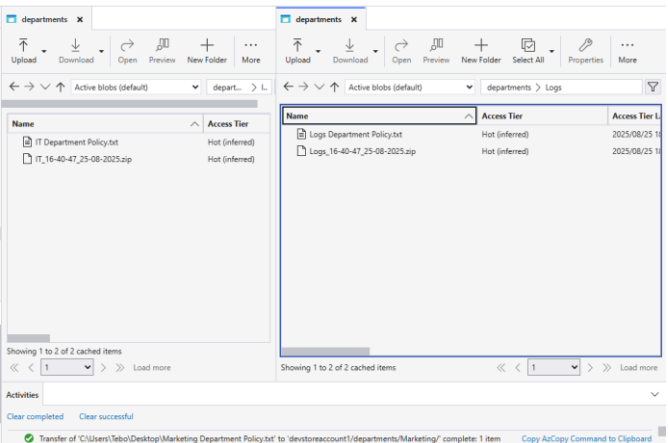
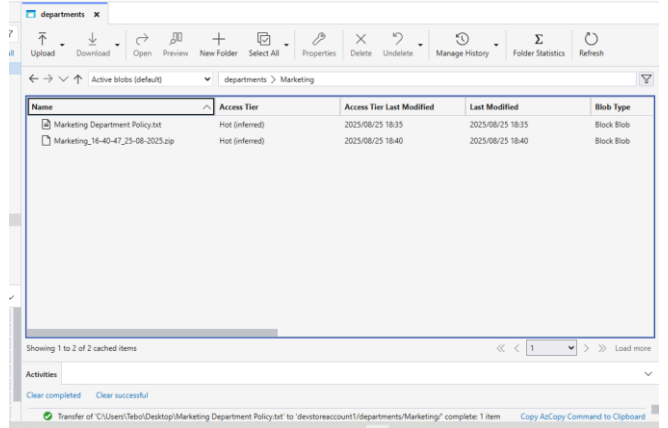
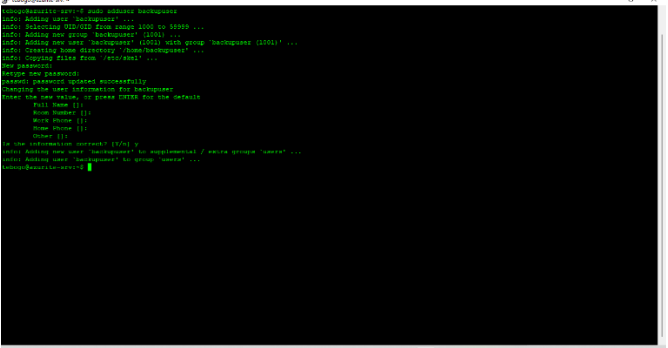
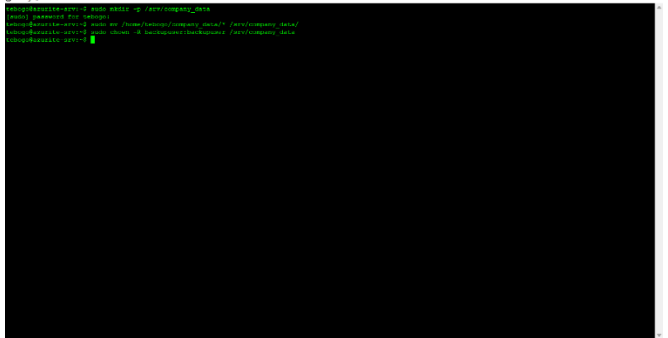
Backup is successful

Run script and verify files are correctly uploaded



Verify Backup



	
	
<p>Create Backupuser</p> <p>Create a dedicated user for running backup scripts</p>	
<p>Move script and hand over ownership</p> <p>Give the new user ownership of the backup script</p>	

```
sudo chown
backupuser:backupuser
```

Update Script

Verify Backup User can
use script

Verify Backup

Crontab and schedule backup

Schedule automatic backups

[illegible][illegible]

```
Job 0dbrf42b3-14dc-0345-8865-cebd84dfffdd0 summary
Elapsed Time (Minutes): 0.034
Number of File Transfers: 1
Number of Folder Property Transfers: 0
Number of Symlink Transfers: 0
Total Number of Transfers: 1
Number of File Transfers Completed: 1
Number of Folder Transfers Completed: 0
Number of File Transfers Failed: 0
Number of Folder Transfers Failed: 0
Number of File Transfers Skipped: 0
Number of Folder Transfers Skipped: 0
Number of Symbolic Links Skipped: 0
Number of Hardlinks Converted: 0
Number of Special Files Skipped: 0
Total Number of Bytes Transferred: 780
Final Job Status: Completed

All uploads completed!
backupuser@azurite-srv:/srv/backup_scripts$
```

[illegible]

The screenshot shows a Windows terminal window with the following commands and output:

```

C:\Users\user> docker run --name myapp -d myapp:latest
c717b260-207b-4070-8070-707070707070
C:\Users\user> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED       STATUS       PORTS
c717b260-207b-4070-8070-707070707070   myapp:latest   "/bin/sh -c 'python3 -m http.server 80'"   2 minutes ago   Up 2 minutes   80/tcp
C:\Users\user> docker logs -f myapp
[{"text": "Hello, World!"}]

```

The terminal output indicates that the container 'myapp' was successfully started and is running. The 'docker ps' command shows the container's ID, image, command, and status. The 'docker logs -f myapp' command shows the output of the container, which is a JSON object containing the text 'Hello, World!'.

Force a manual run to confirm cron executes the script

Force a manual run to confirm cron executes the script

```

background-protect.exe -h
    -h: This task is intended to be run by cron.

Each task in cron has to be defined through a simple line
containing with different fields which the task will be run
and what command to run for the task.

To define the task you can provide comments before the
command. For example, if you want to backup the /home
and /var of the system (don't do this !! in these fields "for"!).

When the task is created it will be stored based on the system's
cron's notion of time and scheduling.

Output of the cron task (including errors) is sent through
email to the user or directly to the log to indicate problems.

For example, you can run a backup of all your user accounts
on a weekly basis with:
# * * * * * tar -czf /var/backups/home.tar /home/

For more information see the manual page of crontab(1) and cron(1)

# * * * * * command
-----
Backup User Schedule Explanation:
Minutes * ~ At * minutes
Hour * ~ At Hour (midnight)
Day of Month * ~ Every day of the month
Day of Week * ~ Every day of the week
Day of Week 5 ~ Friday (0 = Sunday, 5 = Friday)
* ~ Run the command every day on midnight

# * * * * * /usr/bin/cronbackup backup.sh > /var/backups_app/backup.log 2>&1
# * * * * * backup_scripts/app/0015.sh UTC
# /usr/bin/cronbackup_scripts/backup.sh > /var/backups_app/cron_log/backup.log

```

Crontab worked

[illegible]

Done

departments

Upload Download Open Preview New Folder Select All Properties Delete Undo Manage History Folder Statistics Refresh

Active blobs (default) > departments > IT

Name	Access Tier	Access Tier Last Modified	Last Modified	Blob Type
IT Department Policy list	Hot (inferred)	2025/08/25 18:31	2025/08/25 18:31	Block Blob
IT_07-04-13-26-08-2025.zip	Hot (inferred)	2025/08/26 09:54	2025/08/26 09:54	Block Blob
IT_08-15-02-26-08-2025.zip	Hot (inferred)	2025/08/26 10:15	2025/08/26 10:15	Block Blob
IT_16-40-47-25-08-2025.zip	Hot (inferred)	2025/08/25 18:40	2025/08/25 18:40	Block Blob

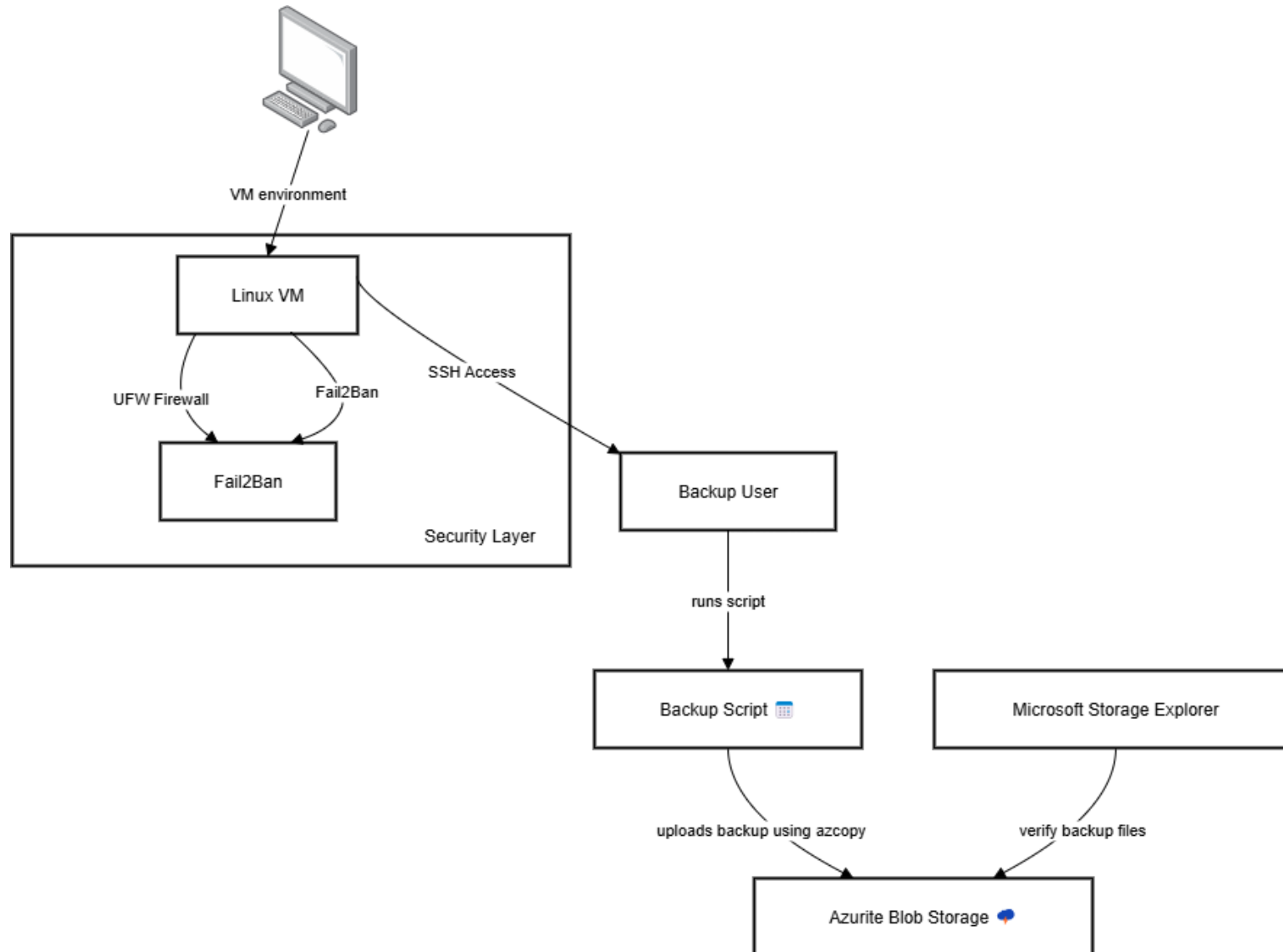
Showing 1 to 4 of 4 cached items

Activities

Clear completed Clear successful

Demonstration

This project is a working demo of how to secure a Linux VM and automate backups to Azurite Blob Storage. It shows the full process from hardening the server, configuring SSH, and setting up Fail2Ban, to running backup scripts with AzCopy and automating them with cron.



Improvements

Use Real Azure Storage

While Azurite is great for testing locally, using an actual Azure Blob Storage account would make this setup cloud-ready. This would also allow testing with production features like redundancy, access tiers, and integration with other Azure services.

Encrypt Backups

Adding an encryption layer ensures that even if data is intercepted or the storage account is compromised, files remain secure. This could be done with tools like GPG, OpenSSL, or even Azure Key Vault for enterprise-grade key management.

Monitoring & Alerts

Right now, the only way to confirm backups is through manual checks in Microsoft Storage Explorer. A better approach would be to add automated logging and email/SMS alerts. For example, if a backup fails or the script doesn't run, the system could notify the administrator immediately.

Infrastructure as Code

Instead of manually configuring the VM, firewall, SSH, and backup scripts, tools like Ansible or Terraform could be used to automate the entire setup. This would make the project repeatable, scalable, and easier to manage across multiple machines or environments.