# Authenticate to Cockpit from anywhere

## Cockpit what?

Cockpit what?

- Interactive Server admin web interface
- Easy setup and troubleshooting for one or a few machines
- Included in all major distros

- Conceptually: Linux session running in a web browser; technically very similar to ssh/VT/GNOME login
- Tool for experimenting, learning, troubleshooting, and doing infrequent tasks

## Extending LVM

Extending LVM

```
pvcreate /dev/sdb2
vgextend vg0 /dev/sdb2
lvresize --extents '+100%FREE' vg0/data1
resize2fs /dev/vg0/data1
```

- for example, adding a new PV to an LVM and resizing the file system you can spend some time coming up with these commands
- lots of possibilities for screwing up
- you can do it simply and safely with Cockpit like this → go to local browser
- Storage page, vg0 in Devices (top right), + in Physical Volumes, add sdb2
- expand data1 table line, click grow

# Authenticate to Cockpit from anywhere

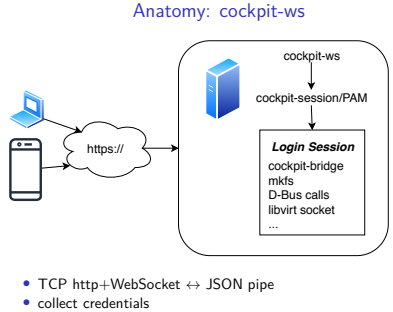
## Accessible from any browser



- being web based makes this server UI available to places that you traditionally don't reach with ssh
- Switch to Windows virt-viewer, open Edge, show Cockpit
- Quit virt-viewer
- Move to local browser, enable mobile mode (Ctrl+Shift+M)
- Zero configuration so far, other than possibly installing cockpit pkg and enabling cockpit.socket
- But wait, you say – want to admin that server over there, but not allowed to open new port and system service?
- In larger environments it's impractical to install cockpit server on hundreds of machines and using the login web page; explain better solutions
- Glimpse of how to customize how cockpit runs and how to authenticate to it

# Authenticate to Cockpit from anywhere

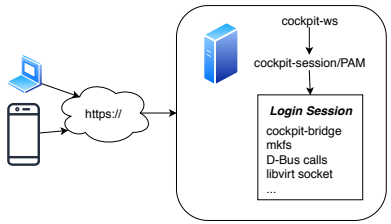
## Anatomy: cockpit-ws




- for configuring, extending, and embedding Cockpit you need to coarsely understand the components of it
- this: default structure, what I just showed you and what you will most probably see the first time you try it
- all components in cockpit communicate to each other via a JSON protocol on standard pipes, usually stdio
- this provides a lot of flexibility and extensibility, as we'll see shortly
- browsers and JS only speak HTTP and WebSocket, and can't directly talk to Linux system APIs
- so you always need a web server somewhere, cockpit-ws
- ws purpose: communicate with the browser for getting credentials: login page, krb negotiation, client cert
- ws: deliver HTML/js content, connects JSON protocol on the WebSocket to pipes to the other components; runs as unprivileged system user

# Authenticate to Cockpit from anywhere

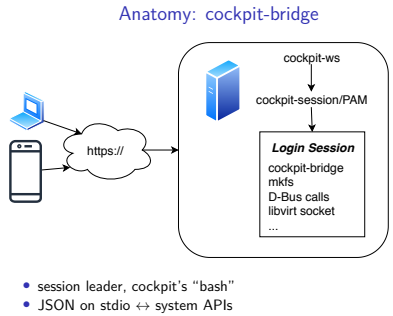
## Anatomy: cockpit-session




- need some helper to actually start session: use creds from ws to start PAM login session, connect pipe to it
- standard is cockpit-session: very small, auditable suid root helper
- but doesn't have to be, that's the flexible part

# Authenticate to Cockpit from anywhere

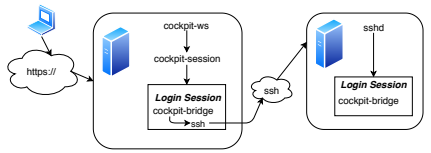
## Anatomy: cockpit-bridge




- bridge: session leader, moral equivalent of what bash is in ssh session
- JSON protocol on stdio to system APIs: exec programs, call D-Bus, work with files or sockets
- runs as target user in login session; complex, but no special privileges

# Authenticate to Cockpit from anywhere

## SSH sessions

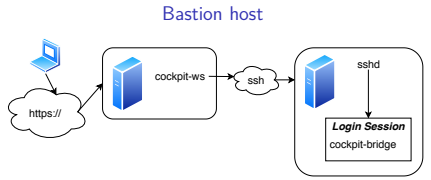



nothing Cockpit specific running outside of the user session

- ws and the login session don't need to run on the same machine
- most obvious replacement of session helper is ssh; that already starts sessions, does the PAM bits and forwards its initial stdio to the session lead; it would just launch cockpit-bridge instead of bash
- browser: go to Dashboard, add cockpit.dev:2201
- interesting property: nothing Cockpit specific running in the system, no ws, no extra open port; only bit is bridge, but that's uninteresting from security POV

# Authenticate to Cockpit from anywhere

## Bastion host




Enforce using ssh in cockpit.conf(5):

```
[WebService]
RequireHost=true
```

- further illustrated by a mode that we call “bastion host”
- disable cockpit-session and local logins, only use ssh
- can run in container
- no ws on critical machines, don’t trust cockpit-session
- switch to browser; log out, use “connect to” for cockpit.dev:2201
- finish the demo script, press Enter

Other authentication setups

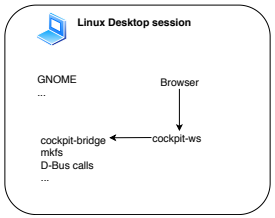
- SSO/Kerberos in Identity Management domains
- smart card/client certificate authentication
- OAuth (external embedding)
- Foreman: included cockpit-ws with dynamic configuration

- Cockpit supports common authentication systems out of the box
- IdM is very common; if you have a krb ticket, you get a session immediately without the login page
- browsers can ask for TLS client certificates, commonly with smart cards, and present them to the web server; latest Cockpit versions supports that
- Foreman has a “Web Console” button; interesting case for seamless transition between Foreman and Cockpit
- Show video
- already has ssh to all maintained machines
- runs a single cockpit-ws process on its server, and dynamically configures it for selected target machine
- custom cockpit session helper to do OAuth between Foreman session and cockpit-ws, and wrap cockpit-ssh session starter
- not enough time to demo and explain all of this; just keep in mind that it’s possible

# Authenticate to Cockpit from anywhere

## Embedding into existing session

Embedding into existing session



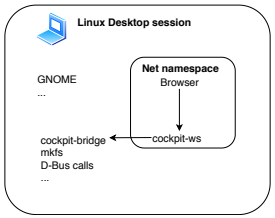

```
cockpit-ws -p 9999 --no-tls --local-session=/usr/bin/cockpit-bridge
firefox http://localhost:9999
```

- what I do want to show: opposite direction; “replace cockpit-session” can also mean “by nothing”
- due to common JSON protocol, we can connect ws directly to a cockpit-bridge
- take a step back: if I want to admin this very machine, it’s in a running Linux session, it knows who I am
- put the whole auth structure inside out and instead run cockpit-ws as my user inside my session
- open –local-session in shell
- open localhost:9999 in firefox
- alarm bells: exposes my session to a TCP port without any auth

# Authenticate to Cockpit from anywhere

## Embedding into existing session: once more with safety!

Embedding into existing session: once more with safety!




/usr/libexec/cockpit-desktop [page]

- need to hide that port; put browser and cockpit-ws into network namespace, then they live in a completely isolated world
- do some work to hide browser chrome, use webkit if available
- cockpit-desktop /
- wants to run priv bridge, can accept or decline
- decline, R/O view
- can show an individual iframe, “page”
- suddenly you end up with a halfway decent desktop app
- just the storage page, replacement for gnome-disks
- cockpit-desktop podman
- cockpit-desktop is small shell script, feel free to inspect and bend to your will

Conclusion
- Authentication is very flexible
- Works with zero configuration
- Can be arbitrarily embedded and customized

- Cockpit provides a set of standard auth protocols that are being used in today's modern deployments
- Once you know about the structure, you can combine ssh, web servers, reverse proxies, and custom auth helpers to embed Cockpit anywhere you want

# Authenticate to Cockpit from anywhere

## Q & A

- Home page leads to mailing lists, documentation
- thanks for your attention; Q+A

Q & A

Contact:

- `#cockpit` on Freenode
- https://cockpit-project.org

Useful links:

- Authentication configuration
- Authentication protocol