# BITCOIN CACHE MACHINE

**Derek Smith**
@farscapian on Twitter
@farscapian on Keybase
farscapian.com

# Bitcoin Cache Machine (BCM)

- Open Source Software w/ MIT License

  https://github.com/BitcoinCacheMachine/BitcoinCacheMachine

- Team Project located on Keybase. Interested? Get involved!

  https://keybase.io/team/btccachemachine

- Disclaimer! BCM is Proof-of-concept stage.
  - Far from feature complete.
  - No formal security evaluations at this time.

# Outline

- Problems/Solutions
- Concept of Operations
- Definitions & Background Info:
  - Software-defined Networking
  - Linux Containers, LXD, and LXC
  - Docker and Docker Swarm
- BCM Technical Architecture
- Demo
- Question/Answers

# Probem

- Existing financial system relies on personal data (fname, lname, SSN, address, phone number, etc) to establish trust relationships.
  - Attackers can open a line-of-credit in your name!
  - High cost associated with fraud prevention.
- Personal data of almost everyone has been hacked numerous times; data is readily purchased on “dark web” using various “cryptocurrencies”.

experian

OPM.GOV

# More problems

- Our personal data resides on computers we don’t control.
  - People are surprised when censorship occurs!
- We live in a state of pervasive mass-surveillance and corporate & financial censorship.
- We constantly leak personal information when using the Internet.
  - SRC/DST IP address in every packet; remote servers & intermediary devices (ISPs, government) knows traffic flows. *metadata=valuable*
  - 3rd party services gain information about you with every REST call & performance suffers due to necessary increase in network latency.
- Internet services today require firewall port forwarding to function. Attackers can scan for open services!! Leaks information=BAD!!

# Solutions

## 1. Use Bitcoin

- Bitcoin is a **censorship-resistant** monetary system and **information registry** with no central authority. Fund security rests on keeping data SECRET.
- Bitcoin does NOT rely on readily available personal information to establish trust.
- Bitcoin expected to be the trusted root anchor for all trustless (i.e., censorship-resistant, permissionless) data.
  - e.g., identity, name/key material, timestamps, etc.
  - OPRETURN and/or Bitcoin-anchored sidechains

# Solutions

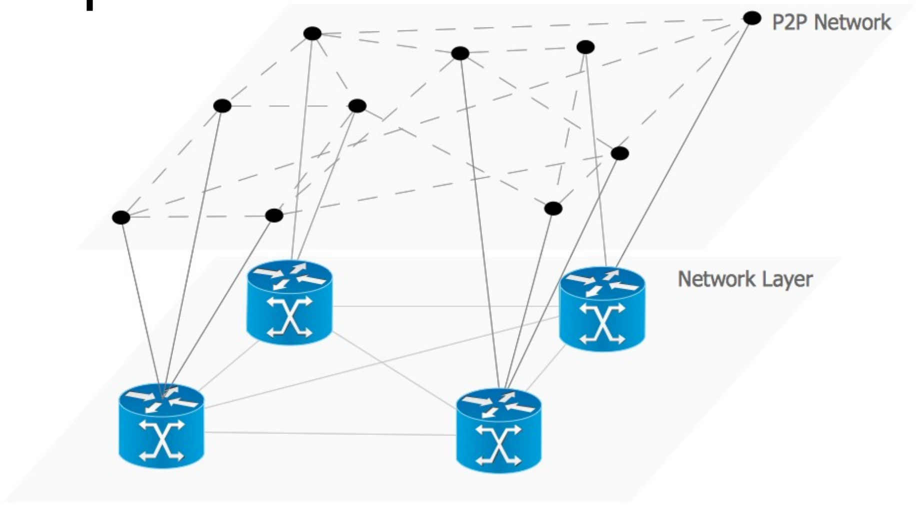
## 2. Use Tor

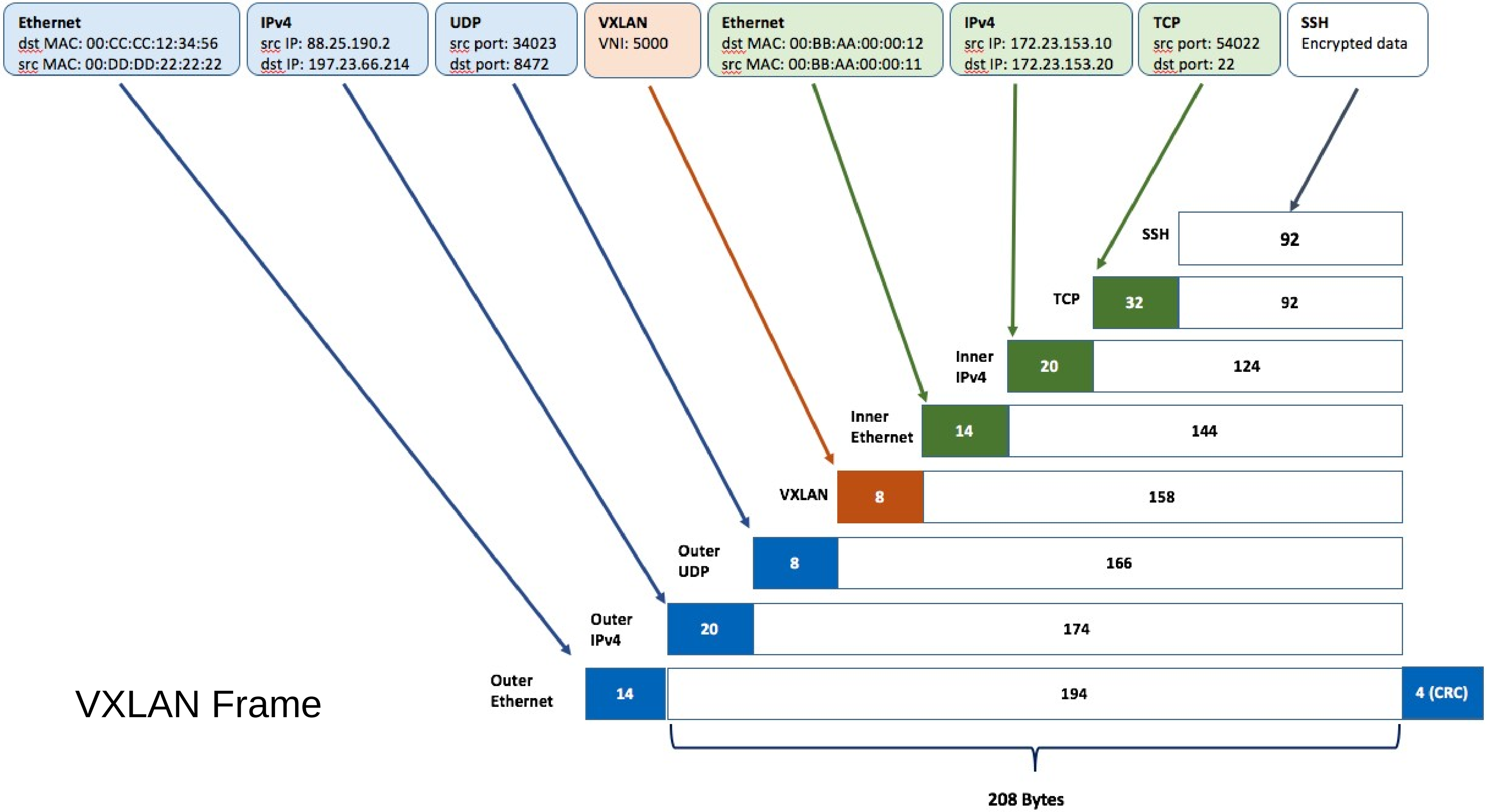
- Provides:
  - – IP anonymity (obscures metadata analysis)
  - – Encryption (combats wiretapping, surveillance)
  - – Layer 4 Authentication* (hides Internet-accessible services from identification during scanning)
- Tor makes it more difficult for public services (e.g., ISPs, NSA) from gaining actionable IP-level traffic flow metadata.
- Tor is an overlay network that rides on top of TCP/IP (the Internet).
  - – Tunnels/overlays achieved with Encapsulation
  - – IP over GRE, IPoGRE, IPinIP, Ethernet in Layer 4 UDP (VXLAN), IPSec tunnel mode, etc.

*Authenticated TOR onion services.

# Overlay Network

- “An overlay network is a computer network that is built on top of another network.”
- Nodes in overlay network are connected by logical links.
- Negatives: protocol overhead, decreased performance due to encapsulation.





**Ethernet**
dst MAC: 00:CC:CC:12:34:56
src MAC: 00:DD:DD:22:22:22

**IPv4**
src IP: 88.25.190.2
dst IP: 197.23.66.214

**UDP**
src port: 34023
dst port: 8472

**VXLAN**
VNI: 5000

**Ethernet**
dst MAC: 00:BB:AA:00:00:12
src MAC: 00:BB:AA:00:00:11

**IPv4**
src IP: 172.23.153.10
dst IP: 172.23.153.20

**TCP**
src port: 54022
dst port: 22

**SSH**
Encrypted data

# VXLAN Frame

# Solution

## 3. Run your own infrastructure

- *“The Cloud”* is a meaningless term
  - typically means ‘other peoples computers’
- Trusted third parties are security holes.

> *“The invocation or assumption in a security protocol design of a "trusted third party" (TTP) or a "trusted computing base" (TCB) controlled by a third party constitutes the introduction of a security hole into that design.”*

- But you can have “*Cloud-y*” technologies available at your home or business!
- BCM allows you to get the benefits of “cloud” technologies without infrastructural middle-men (AWS/Azure/NSA/Level3/AT&T, etc.)
  - But you have to bring your own x86_64

# Software-Defined Networking compared with Legacy IT

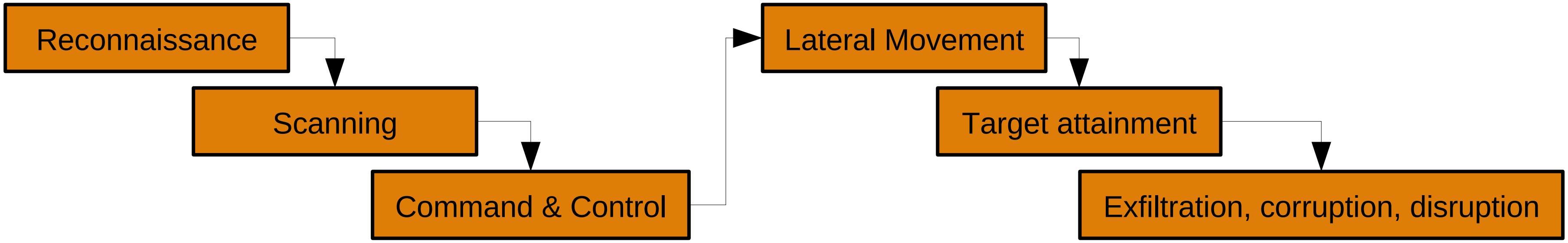
- **Legacy IT Data Centers**
  1. Provisioning is manual.
  2. ABILITY to scale is NOT in lockstep with application & data needs
  3. Physical infrastructure requires specialized devices and network topologies.
  4. Logging into & configuring network and servers via manual configuration changes.
  5. Provision servers or VMs for essential network & security services, e.g., DNS, DHCP, proxies, NAT, etc.
  6. Adding application-level security services: IDS/IPS, Web-application firewall, network content inspection, etc.
  7. Application installed by manually running installation files at console.
  8. Manually configure backup, restoration, disaster recovery, key management.

- **Software-defined (Virtualized Data Center)**
  1. Automated provisioning
  2. ABILITY to scale IS in lockstep with application & data needs
  3. Streamlined Physical Infrastructure – LAN segment & commodity hardware, flat network designs
  4. All changes to infrastructure are designed in code and pushed to production via management interface via continuous integration.
  5. Run container providing essential services & optionally expose physical network underlay.
  6. Operations can focus on adding and maintaining commodity hardware. Initial remote provisioning is made simple!
  7. Attach (system & application) containers (self-contained processes) to networks; treat each container like an individual host with networking and storage.
  8. EVERYTHING (network topology, storage, scale, etc., backup, restoration, scanning, etc.) is scripted and/declared.

# Bitcoin Cache Machine

Bitcoin Cache Machine (BCM) allows you to create a **self-hosted, privacy-preserving, software-defined data-center**.

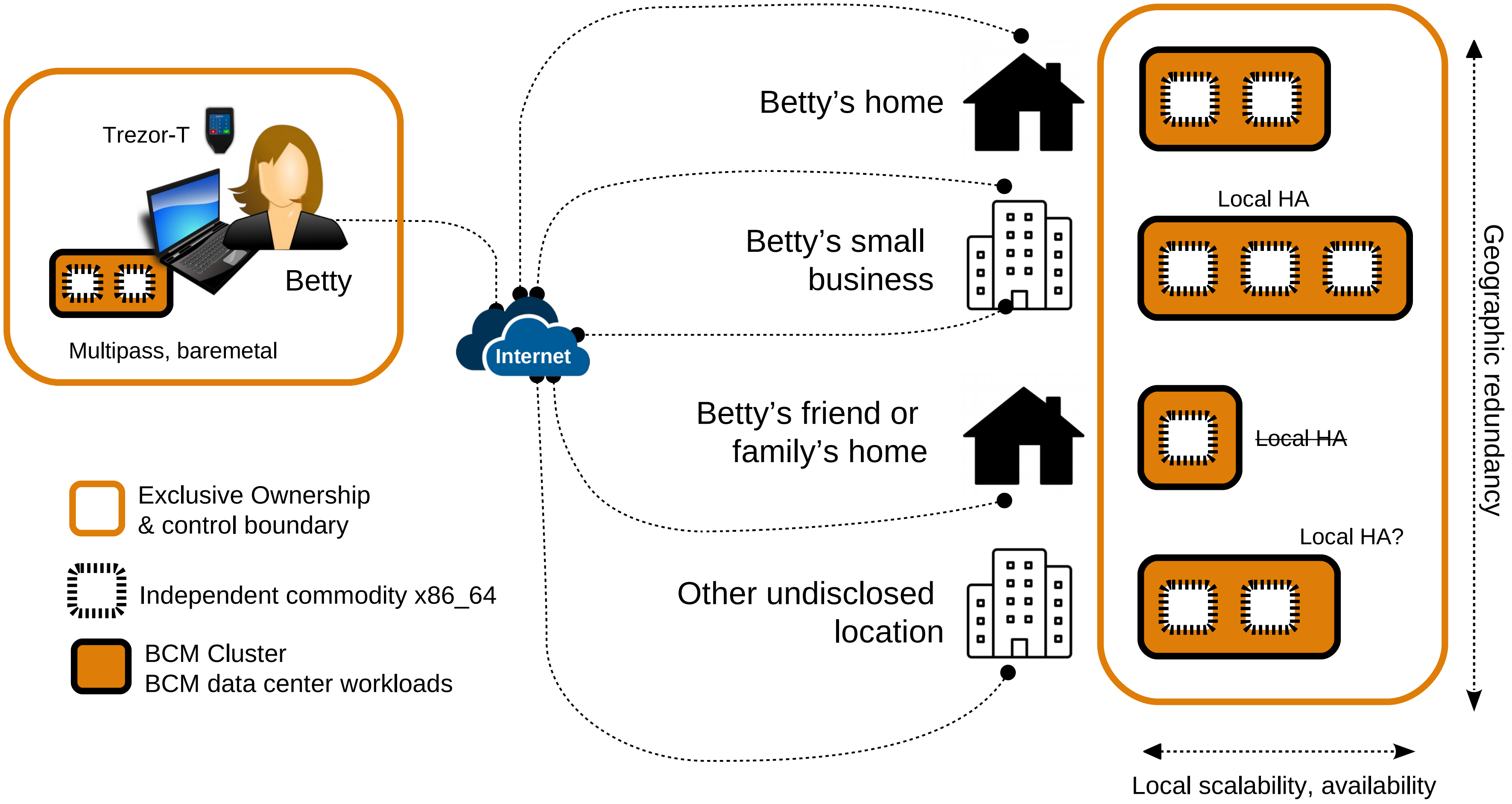
- Goals and Values:
  - Trust minimization → Individual Sovereignty
  - Fundamental right to privacy → Good Security
- BCM helps lower risk of Reconnaissance & Scanning stages of typical cyber attack:



# BCM Development Goals

- Provide a self-contained, distributed, event-driven, software-defined data center that focuses on operational Bitcoin and Lightning-related IT infrastructure.
- Integrate exclusively free and open source software (FOSS).
- Automate the deployment to commodity hardware.
- Automate operational tasks (e.g., backups, updates, scanning, key and password management, etc.) of BCM data centers.
- Require hardware wallets for cryptographic operations (PGP, SSH, and Bitcoin/Lightning transactions).
- Pre-configure all software to protect user's privacy (e.g., TOR for external communication, disk encryption, minimal attack surface, etc.).
- Pursue Global Consensus (bitcoin) and Local Consensus Models (minimally trusted infrastructure) for core platform components.

# BCM Concept of Operations



Trezor-T

Betty

Multipass, baremetal

Internet

Betty's home

Betty's small business

Betty's friend or family's home

Other undisclosed location

Local HA

~~Local HA~~

Local HA?

Geographic redundancy

Local scalability, availability

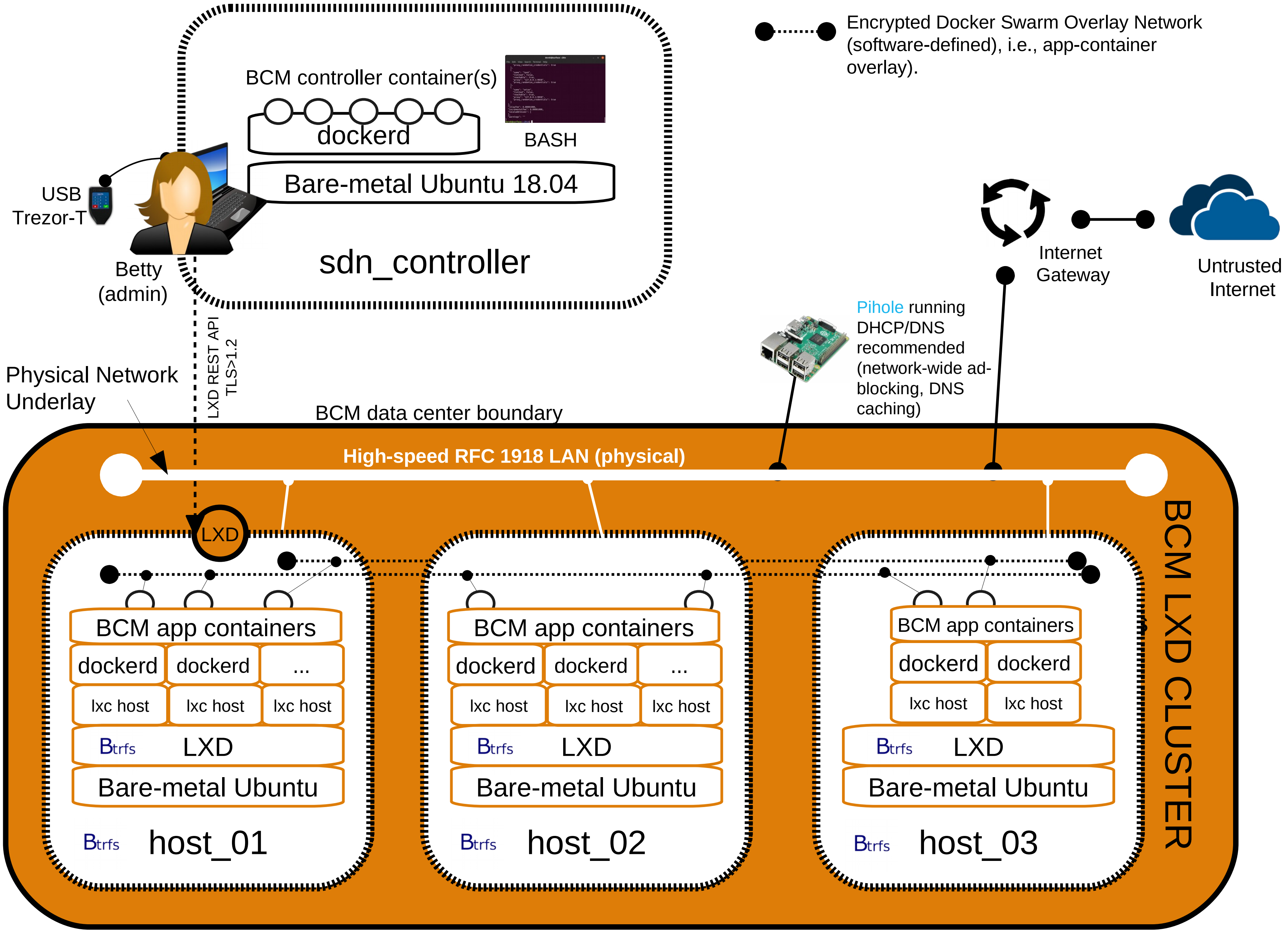
□ Exclusive Ownership & control boundary

⬚ Independent commodity x86_64

■ BCM Cluster
BCM data center workloads

# Definitions

- SDN Controller
  - manages one or more LXD Clusters (where your data center workloads actually run) via LXD REST API.
  - SDN Controller is a user-facing laptop/desktop.
- LXD Cluster
  - collection of **one or more** LXD endpoints with a private networking environment that is low latency and high bandwidth, such as a **home or office LAN**.
- LXD Endpoint:
  - LXD process running on independent hardware failure domains (x86_64).


Encrypted Docker Swarm Overlay Network (software-defined), i.e., app-container overlay).
BCM controller container(s)
dockerd
BASH
Bare-metal Ubuntu 18.04
sdn_controller
USB
Trezor-T
Betty
(admin)
Internet
Gateway
Untrusted
Internet
Pihole running DHCP/DNS recommended (network-wide ad-blocking, DNS caching)
LXD REST API
TLS>1.2
Physical Network
Underlay
BCM data center boundary
High-speed RFC 1918 LAN (physical)
LXD
BCM app containers
dockerd
dockerd
...
lxc host
lxc host
lxc host
Btrfs
LXD
Bare-metal Ubuntu
Btrfs
host_01
BCM app containers
dockerd
dockerd
...
lxc host
lxc host
lxc host
Btrfs
LXD
Bare-metal Ubuntu
Btrfs
host_02
BCM app containers
dockerd
dockerd
lxc host
lxc host
Btrfs
LXD
Bare-metal Ubuntu
Btrfs
host_03
BCM LXD CLUSTER

# Demo – Git the BCM Repo

1. Start with a fresh Ubuntu 18.04 LTS installation on your laptop/desktop.
2. Visit BCM git repo using TOR browser & read:
   https://github.com/BitcoinCacheMachine/BitcoinCacheMachine
3. Follow “Getting Started” section

```
sudo apt-get update
sudo apt-get install -y tor git
BCM_GITHUB_REPO_URL="https://github.com/BitcoinCacheMachine/BitcoinCacheMachine"
git config --global http.$BCM_GITHUB_REPO_URL.proxy socks5://localhost:9050
```

```
BCM_GIT_DIR="$HOME/git/github/bcm"
mkdir -p "$BCM_GIT_DIR"
git clone "$BCM_GITHUB_REPO_URL" "$BCM_GIT_DIR"
cd "$BCM_GIT_DIR"
./setup.sh
source "$HOME/.bashrc"
```

4. Run the Demo app (CLI example).

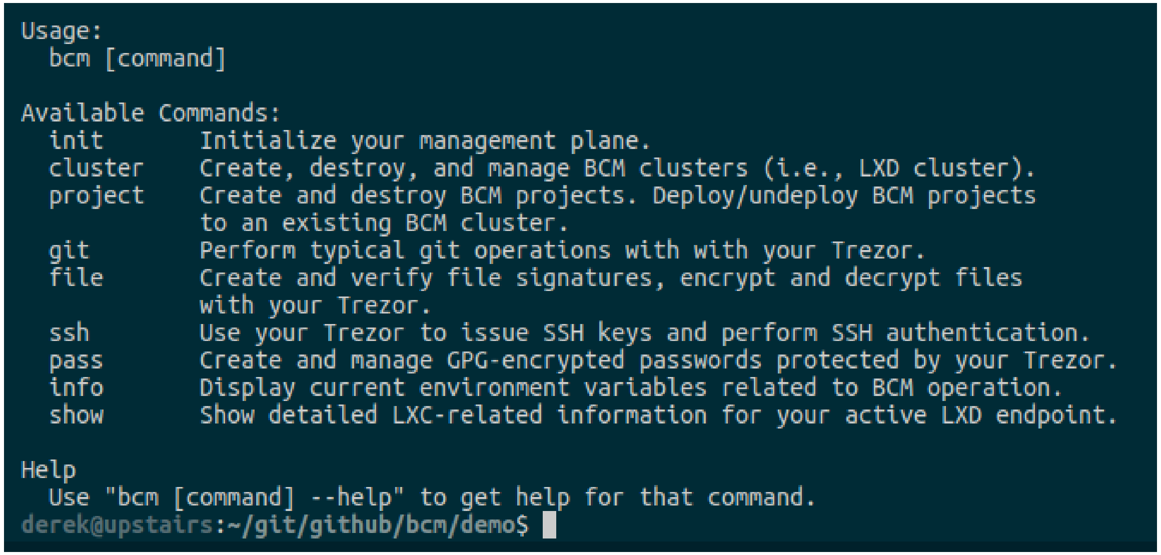
```
bash -c "$BCM_GIT_DIR/demo/meet_up.sh"
```

SDN Controller

TODO: move the trusted BCM git repo to a onion service or IPFS pub/sub. Trust in 3rd parties—including Github--is BAD!

# User Interface: BCM CLI

- BCM comes with a CLI; used at SDN Controller.
- SDN Controller docker image includes Trezor integration.
  - BCM CLI (just a bunch of BASH scripts) detects new Trezor USB device then mounts it into SDN controller docker container in interactive foreground mode.

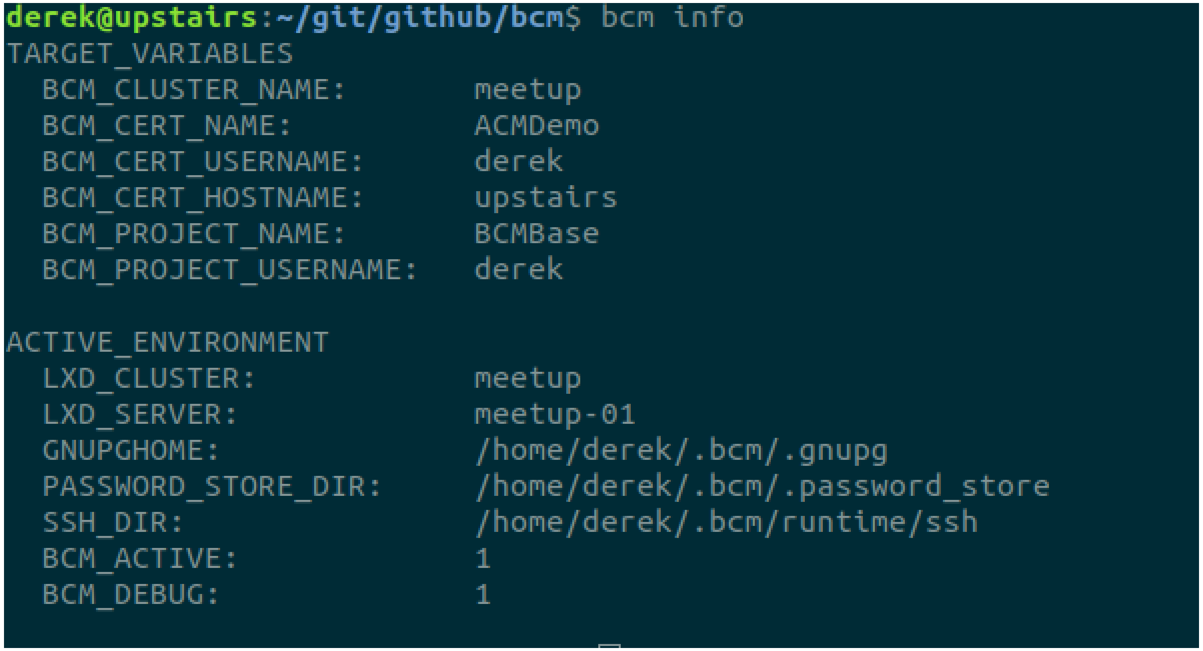


```
Usage:
  bcm [command]

Available Commands:
  init       Initialize your management plane.
  cluster    Create, destroy, and manage BCM clusters (i.e., LXD cluster).
  project    Create and destroy BCM projects. Deploy/undeploy BCM projects
             to an existing BCM cluster.
  git        Perform typical git operations with with your Trezor.
  file       Create and verify file signatures, encrypt and decrypt files
             with your Trezor.
  ssh        Use your Trezor to issue SSH keys and perform SSH authentication.
  pass       Create and manage GPG-encrypted passwords protected by your Trezor.
  info       Display current environment variables related to BCM operation.
  show       Show detailed LXC-related information for your active LXD endpoint.

Help
  Use "bcm [command] --help" to get help for that command.
derek@upstairs:~/git/github/bcm/demo$ 
```

SDN Controller

# `bcm info`

- Shows your active BCM environment variables.
- When BCM_ACTIVE=0, the BCM CLI switches to your `~/.gnupg`, `~/.password_store`, and `~/.ssh` stores.
    - Allows you to use Trezor outside of BCM data-center context.
    - Sign public git commit using different identity. Use passphrases to separate GPG identity trust boundary
- Use when managing your BCM clusters and deployed projects!

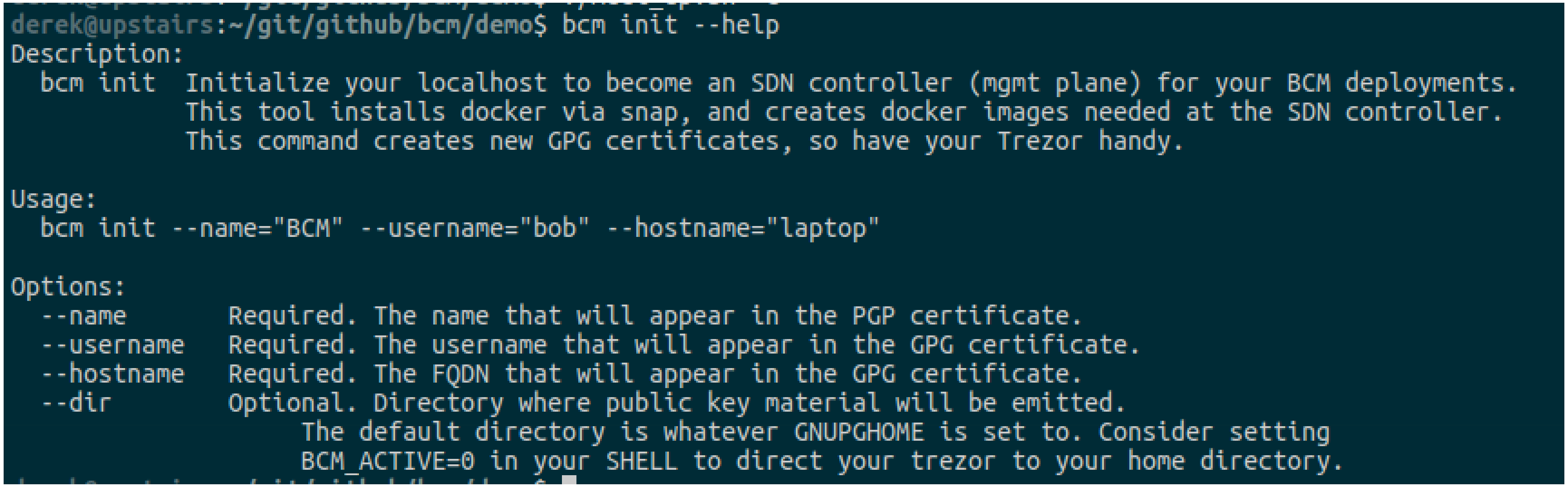


```
derek@upstairs:~/git/github/bcm$ bcm info
TARGET_VARIABLES
  BCM_CLUSTER_NAME:          meetup
  BCM_CERT_NAME:             ACMDemo
  BCM_CERT_USERNAME:         derek
  BCM_CERT_HOSTNAME:         upstairs
  BCM_PROJECT_NAME:          BCMBase
  BCM_PROJECT_USERNAME:      derek

ACTIVE_ENVIRONMENT
  LXD_CLUSTER:               meetup
  LXD_SERVER:                meetup-01
  GNUPGHOME:                 /home/derek/.bcm/.gnupg
  PASSWORD_STORE_DIR:        /home/derek/.bcm/.password_store
  SSH_DIR:                   /home/derek/.bcm/runtime/ssh
  BCM_ACTIVE:                1
  BCM_DEBUG:                 1
```

SDN Controller

# bcm init

- Makes sure your SDN controller has all the software it needs
- Generates new trezor-backed GPG certificates & password store.
- Provisions ~/.bcm and all subdirectories (encfs)

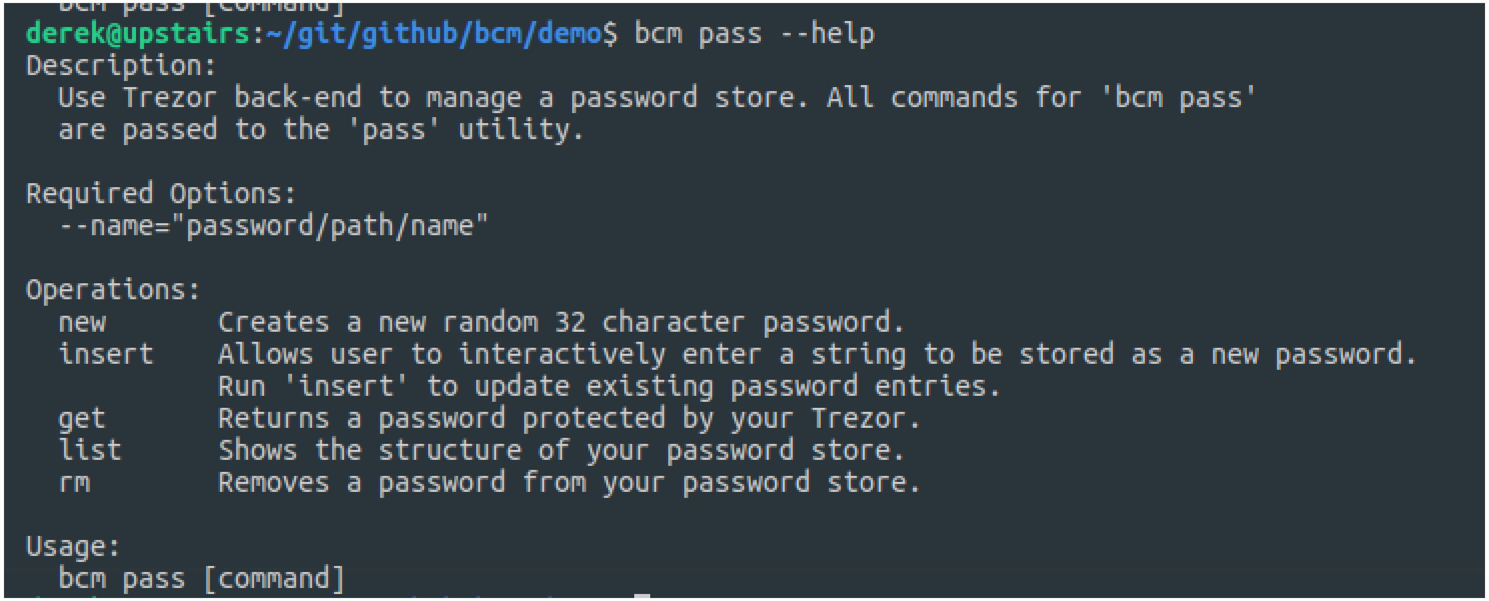


```
derek@upstairs:~/git/github/bcm/demo$ bcm init --help
Description:
  bcm init  Initialize your localhost to become an SDN controller (mgmt plane) for your BCM deployments.
            This tool installs docker via snap, and creates docker images needed at the SDN controller.
            This command creates new GPG certificates, so have your Trezor handy.

Usage:
  bcm init --name="BCM" --username="bob" --hostname="laptop"

Options:
  --name        Required. The name that will appear in the PGP certificate.
  --username    Required. The username that will appear in the GPG certificate.
  --hostname    Required. The FQDN that will appear in the GPG certificate.
  --dir         Optional. Directory where public key material will be emitted.
                     The default directory is whatever GNUPGHOME is set to. Consider setting
                     BCM_ACTIVE=0 in your SHELL to direct your trezor to your home directory.
```

SDN Controller

# bcm pass --help

- SDN Controller Container uses the Standard Unix Password Manager
- BCM CLI is a wrapper around ‘pass’
- Not all ‘pass’ operations are exposed



```
  bcm pass [command]
derek@upstairs:~/git/github/bcm/demo$ bcm pass --help
Description:
  Use Trezor back-end to manage a password store. All commands for 'bcm pass'
  are passed to the 'pass' utility.

Required Options:
  --name="password/path/name"

Operations:
  new       Creates a new random 32 character password.
  insert    Allows user to interactively enter a string to be stored as a new password.
            Run 'insert' to update existing password entries.
  get       Returns a password protected by your Trezor.
  list      Shows the structure of your password store.
  rm        Removes a password from your password store.

Usage:
  bcm pass [command]
```

SDN Controller

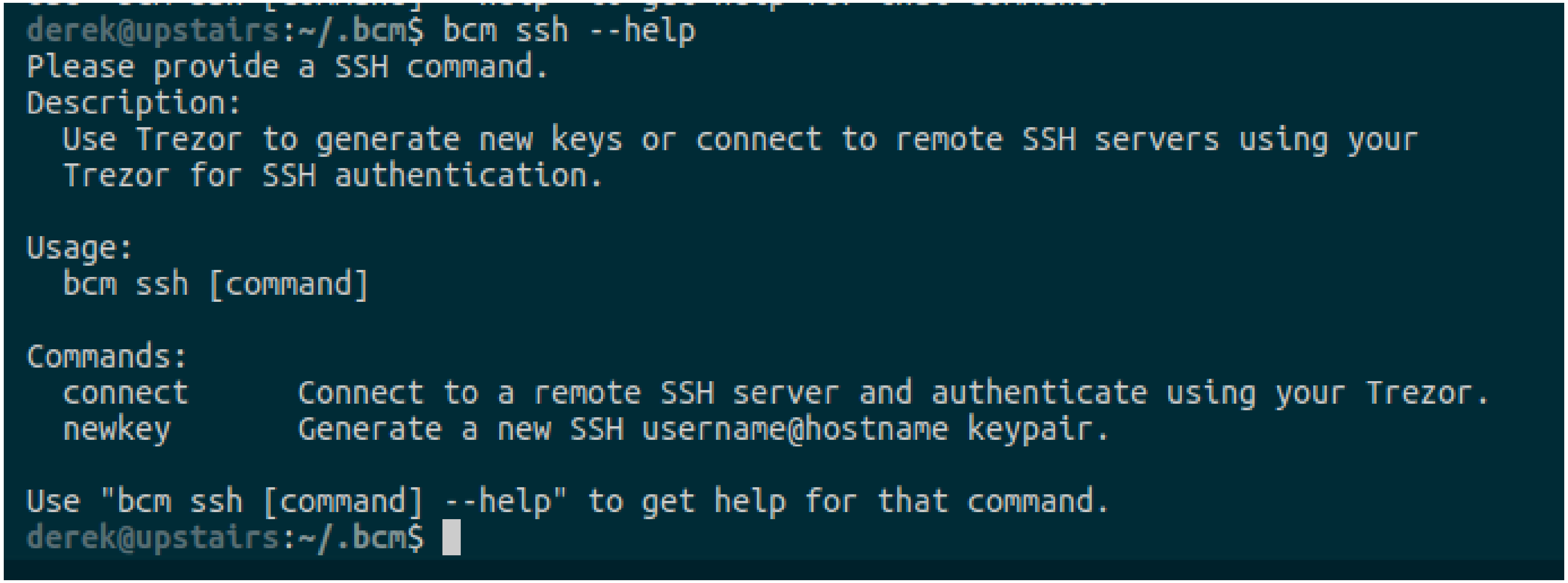
# Trezor-based password management

- Seed phrase/mnemonic - BIP0039
  - Words typically written down representing your cryptographic seed.
  - Can use SSS, encrypted offsite backups, steganography, to protect seed.
- Type 2 Deterministic Wallets – BIP0032
  - All addresses, accounts, etc., are derived from cryptographic seed.
  - Separate public & private-key generation.
  - Read-only (e.g., xpub required for account balances) and write (xpriv) (per derivation path). XPRIVs ALWAYS stays on Trezor!
- Passphrases - BIP0038
  - Additional words entered post-PIN authentication; drops you into a completely different root wallet.
  - Good OPSEC; protects against $5 wrench attack.
  - BCM: Use passphrases for personas (private/public/business) or runtime environments (dev, staging, production), etc.

SDN Controller

# bcm ssh --help

- Unique identities user@hostname PER BIP0032 path
- Hostname MUST be DNS resolvable by the SDN controller (use *etc/hosts or the DNS)*
- *Username MUST a provisioned user on Ubuntu client (default ‘ubuntu’)*
- *Only works on localnet atm, TORified bcm ssh connect planned.*

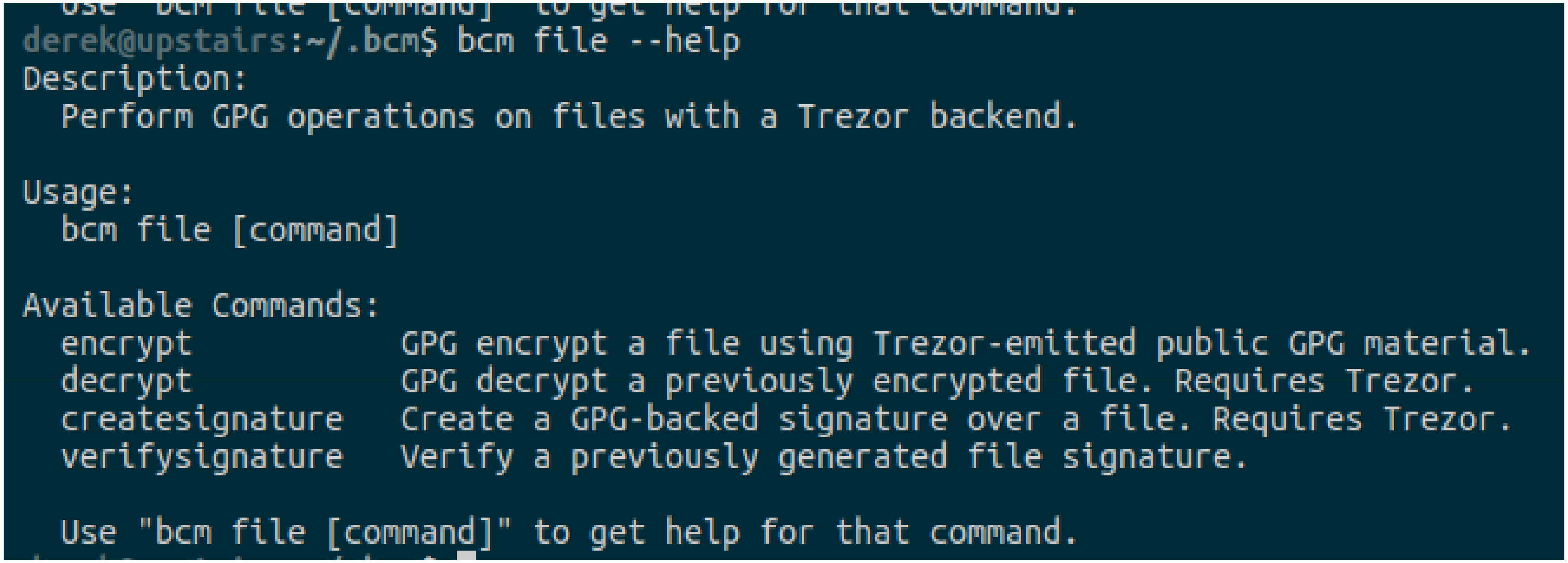


```
derek@upstairs:~/.bcm$ bcm ssh --help
Please provide a SSH command.
Description:
  Use Trezor to generate new keys or connect to remote SSH servers using your
  Trezor for SSH authentication.

Usage:
  bcm ssh [command]

Commands:
  connect      Connect to a remote SSH server and authenticate using your Trezor.
  newkey       Generate a new SSH username@hostname keypair.

Use "bcm ssh [command] --help" to get help for that command.
derek@upstairs:~/.bcm$ 
```

SDN Controller

# bcm file --help

*Use BCM to encrypt/decrypt arbitrary files via GPG using your trezor*
*BCM GPG can be used to create/verify file signatures.*



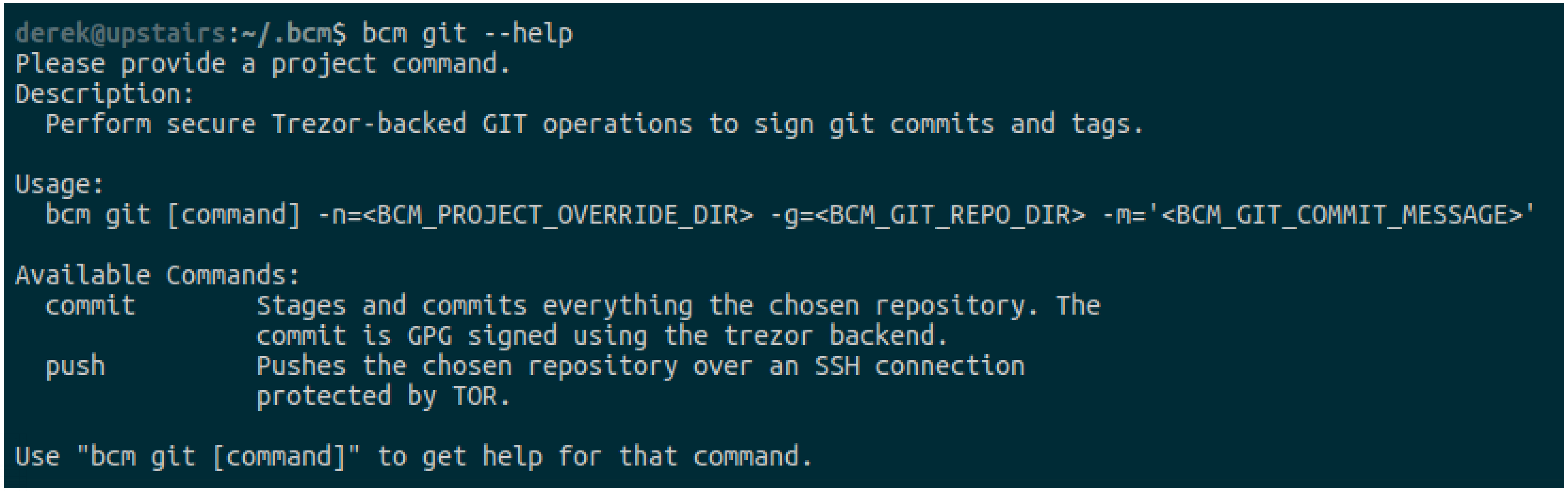
```
  Use "bcm file [command]" to get help for that command.
derek@upstairs:~/.bcm$ bcm file --help
Description:
  Perform GPG operations on files with a Trezor backend.

Usage:
  bcm file [command]

Available Commands:
  encrypt            GPG encrypt a file using Trezor-emitted public GPG material.
  decrypt            GPG decrypt a previously encrypted file. Requires Trezor.
  createsignature    Create a GPG-backed signature over a file. Requires Trezor.
  verifysignature    Verify a previously generated file signature.

  Use "bcm file [command]" to get help for that command.
```

SDN Controller

# bcm git --help

*Use Trezor to sign git commits and git tags.*

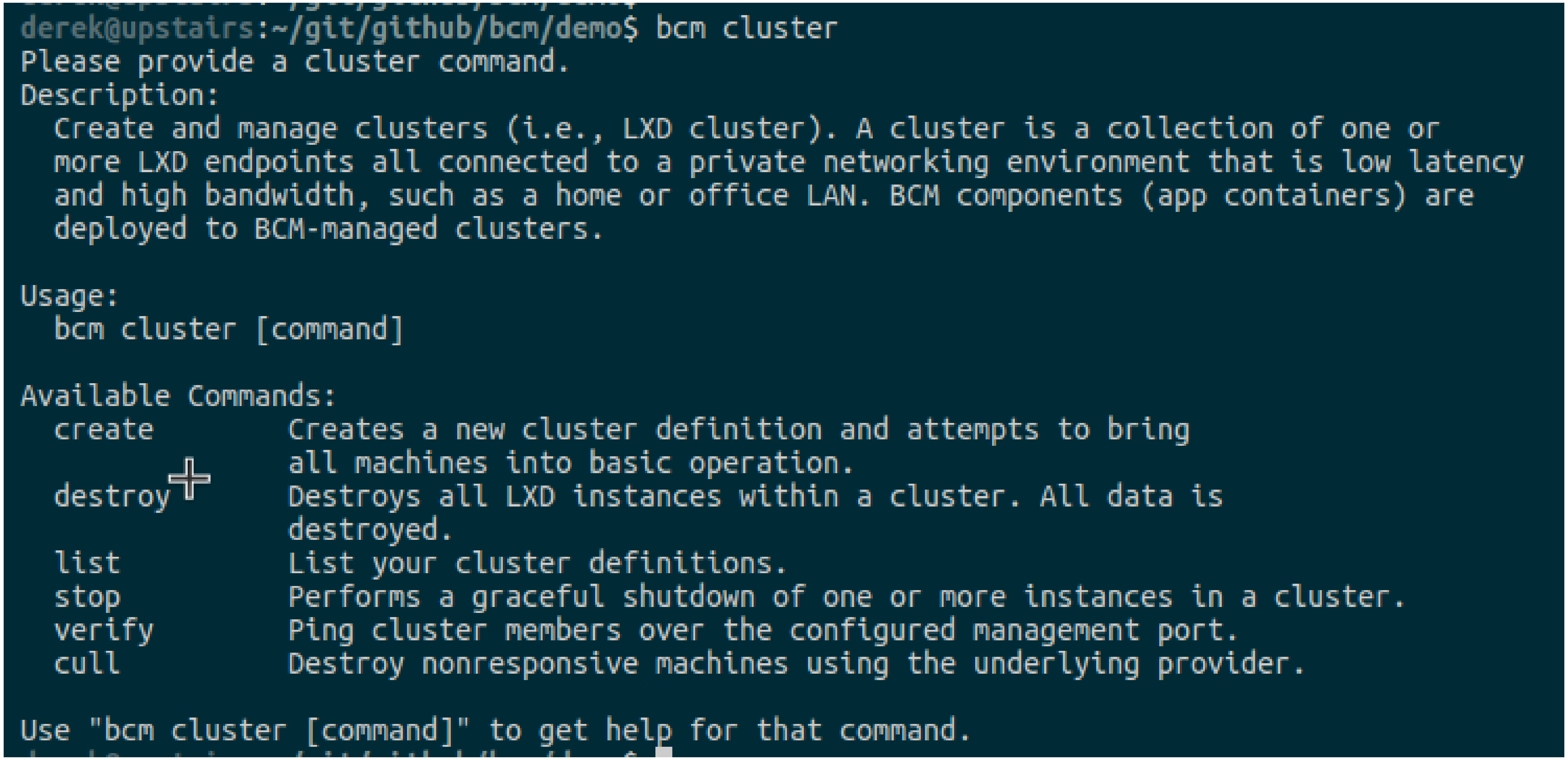
- *BCM PLANNED features*
  - *Git push to remote repositories. Transport is SSH over TOR. Authentication is SSH using Trezor.*
  - *BCM data centers will (eventually) host Git repo servers for backups.*



```
derek@upstairs:~/.bcm$ bcm git --help
Please provide a project command.
Description:
  Perform secure Trezor-backed GIT operations to sign git commits and tags.

Usage:
  bcm git [command] -n=<BCM_PROJECT_OVERRIDE_DIR> -g=<BCM_GIT_REPO_DIR> -m='<BCM_GIT_COMMIT_MESSAGE>'

Available Commands:
  commit        Stages and commits everything the chosen repository. The
                commit is GPG signed using the trezor backend.
  push          Pushes the chosen repository over an SSH connection
                protected by TOR.

Use "bcm git [command]" to get help for that command.
```

SDN Controller

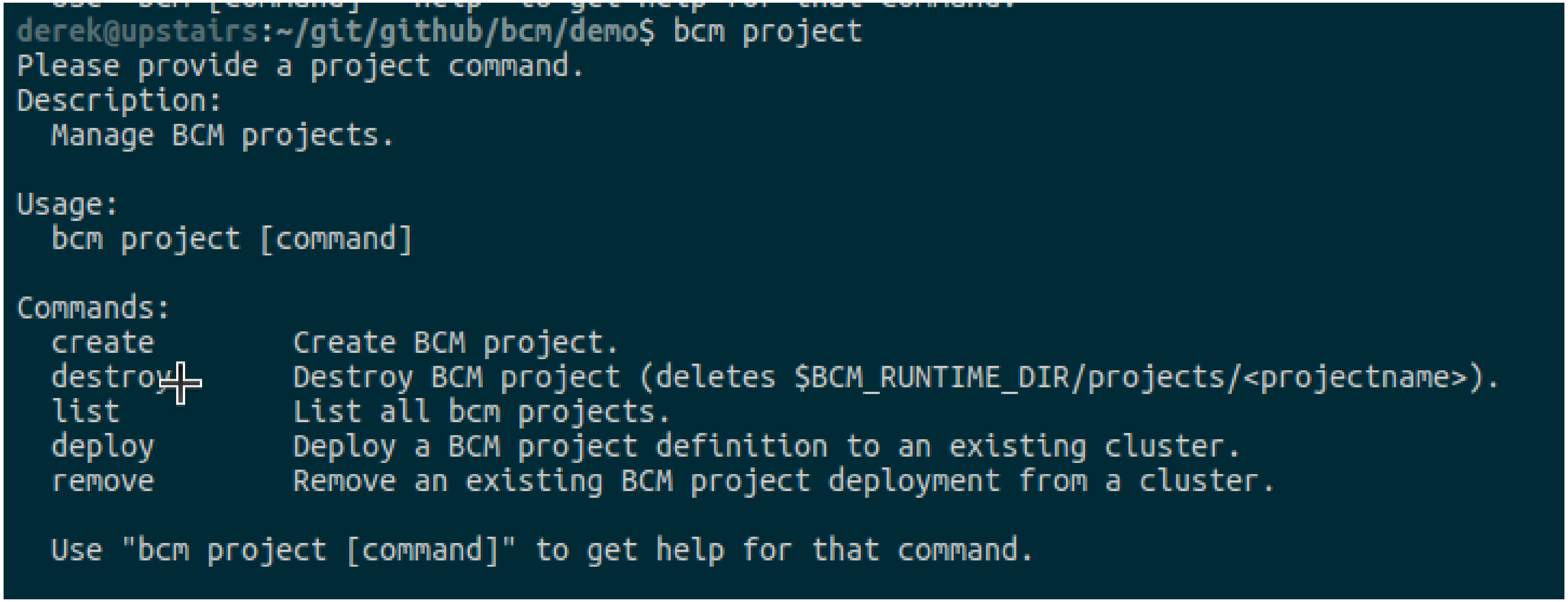
# bcm cluster --help



```
derek@upstairs:~/git/github/bcm/demo$ bcm cluster
Please provide a cluster command.
Description:
  Create and manage clusters (i.e., LXD cluster). A cluster is a collection of one or
  more LXD endpoints all connected to a private networking environment that is low latency
  and high bandwidth, such as a home or office LAN. BCM components (app containers) are
  deployed to BCM-managed clusters.

Usage:
  bcm cluster [command]

Available Commands:
  create        Creates a new cluster definition and attempts to bring
                all machines into basic operation.
  destroy       Destroys all LXD instances within a cluster. All data is
                destroyed.
  list          List your cluster definitions.
  stop          Performs a graceful shutdown of one or more instances in a cluster.
  verify        Ping cluster members over the configured management port.
  cull          Destroy nonresponsive machines using the underlying provider.

Use "bcm cluster [command]" to get help for that command.
```

SDN Controller

# bcm project --help



```
  Use "bcm [command] --help" to get help for that command.
derek@upstairs:~/git/github/bcm/demo$ bcm project
Please provide a project command.
Description:
  Manage BCM projects.

Usage:
  bcm project [command]

Commands:
  create        Create BCM project.
  destroy       Destroy BCM project (deletes $BCM_RUNTIME_DIR/projects/<projectname>).
  list          List all bcm projects.
  deploy        Deploy a BCM project definition to an existing cluster.
  remove        Remove an existing BCM project deployment from a cluster.

  Use "bcm project [command]" to get help for that command.
```

SDN Controller

# BCM SDN Controller

- SDN controller (user-facing laptop/desktop) deploys and (re-)configures data center(s) via the SSH & LXD API.
- BCM workloads (app-level containers) (bitcoind/lightning/kafka) are scheduled across a set of docker daemons distributed across LXD endpoints.
- TLS over TOR (authenticated onion service) PLANNED for remote data center management plane interface.
- BCM data center workloads MAY be run on BCM Controller (LXD service endpoint at 127.0.10.1).
  - Great for deploying a fully functional full node infrastructure to your laptop! (standalone mode)
  - Provides NO Local-HA*.

SDN Controller

# SDN Controller Files “`$HOME/.bcm`”

All GIT repos!

Trezor-generated GPG certificates. Protects password store, signs other certificates, acts as trusted root CA.

Trezor-backed password management using pass.

EncFS (FUSE) encrypted file system in user space. Encrypts all files in `./runtime/*`. Encryption passwords stored in `./.pass/bcm`

Unencrypted view of data-at-rest under FUSE mount

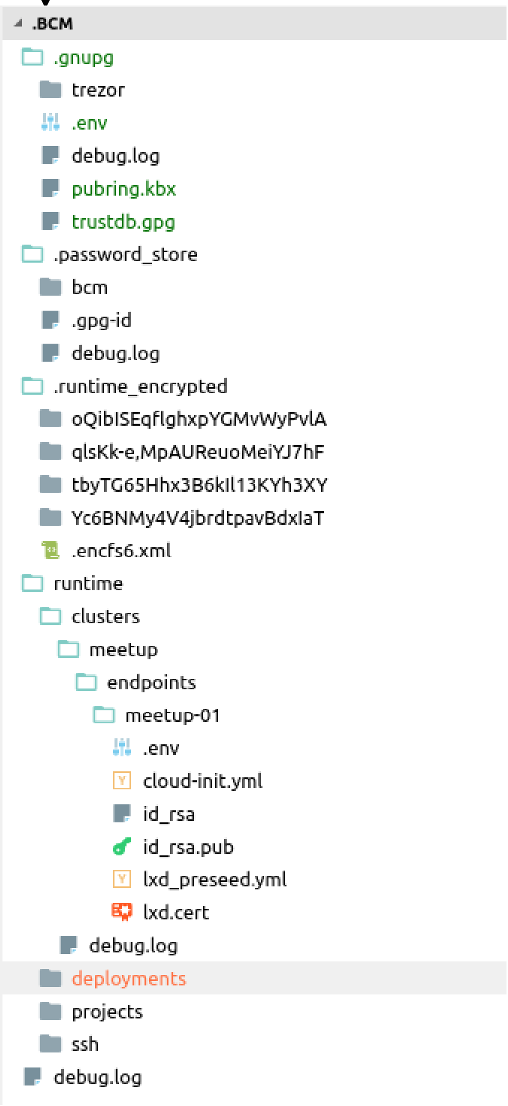
Your deployed clusters:

List of LXD endpoints in your cluster.

List project deployments (projects deployed to clusters).

BCM Project Definitions (containerized software stack).

Trezor-backed SSH public key identities.



```
.BCM
  .gnupg
    trezor
    .env
    debug.log
    pubring.kbx
    trustdb.gpg
  .password_store
    bcm
    .gpg-id
    debug.log
  .runtime_encrypted
    oQibISEqflghxpYGMvWyPvlA
    qlsKk-e,MpAUReuoMeiYJ7hF
    tbyTG65Hhx3B6kIl13KYh3XY
    Yc6BNMy4V4jbrdtpavBdxIaT
    .encfs6.xml
  runtime
    clusters
      meetup
        endpoints
          meetup-01
            .env
            cloud-init.yml
            id_rsa
            id_rsa.pub
            lxd_preseed.yml
            lxd.cert
      debug.log
    deployments
    projects
    ssh
  debug.log
```

SDN Controller

# A word about multipass

- multipass is a tool that helps you create KVM/QEMU VMs on your dev machine.
- BCM uses cloud-init to automatically provision each Ubuntu VM to receive SSH/LXD commands.
- Use multipass to simulate a multi-endpoint-cluster on a single machine.
  - Used for development only since you don’t get independent failure domains. Currently no multipass network-based API.

# Containers

- Containers allow you to “package and isolate applications with their entire runtime environment—all of the files necessary [dependencies] to run.”
- Containers
  - Help reduce conflicts between operating environments (dev, staging, production); mitigates “but it worked fine on my computer...”
  - Facilitates separation between Operations & App Development (app developers must implement operational concerns, e.g., backups, key mgmt, in code).
  - Ops focuses on providing physical networking, compute, storage hardware.
- BCM uses TWO levels of containerization
  - System Containers – controls hardware resources via LXD.
  - App Containers – provides stateless application functionality via Docker.

```
FROM ubuntu:cosmic

ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update
RUN apt-get install -y apt-utils git pass

# Currently using https://github.com/romanz/trezor-agent https://github.com/romanz/trezor-agent/blob/master/doc/INSTALL.md
RUN apt-get install -y python3-pip python3-dev python3-tk libusb-1.0-0-dev libudev-dev --fix-missing
RUN apt-get install -y wait-for-it openssh-client git tor usbutils curl gnupg2

# 2. Install the TREZOR agent
RUN pip3 install Cython hidapi
RUN pip3 install trezor_agent

RUN mkdir /tmp/trezor
RUN git clone https://github.com/romanz/trezor-agent /tmp/trezor
WORKDIR /tmp/trezor
RUN git checkout latest-trezorlib
RUN pip3 install -e /tmp/trezor/agents/trezor

WORKDIR /gitrepo

# run this script to quickly configure and commit and sign a repo
RUN mkdir /bcm
ADD ./container_scripts/commit_sign_git_repo.sh /bcm/commit_sign_git_repo.sh
RUN chmod +x /bcm/commit_sign_git_repo.sh

# run this script to quickly configure and commit and sign a repo
ADD ./container_scripts/git_push.sh /bcm/git_push.sh
RUN chmod +x /bcm/git_push.sh

# run this script to quickly configure and commit and sign a repo
ADD ./container_scripts/git_pull.sh /bcm/git_pull.sh
RUN chmod +x /bcm/git_pull.sh

# run this script to quickly configure and commit and sign a repo
ADD ./container_scripts/docker-entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

RUN mkdir -p /root/.bcm

ENV GNUPGHOME=/root/.gnupg/trezor
```

# SDN Controller Dockerfile

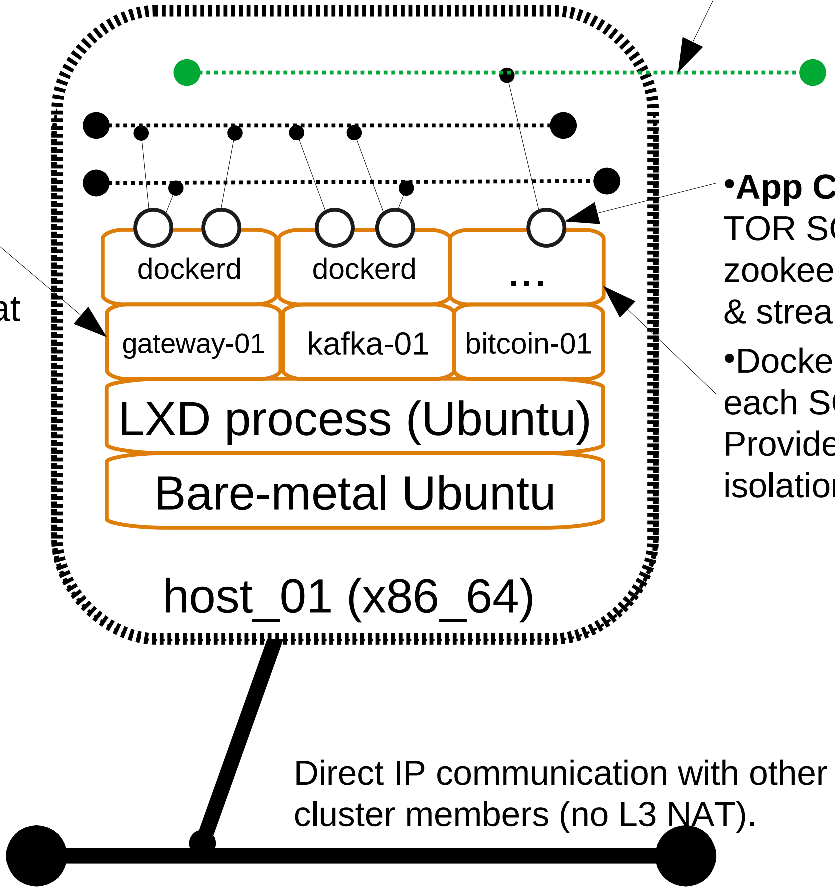
* SDN build process NOT yet protected by TOR.

SDN Controller

# BCM Container Structure

LXC Hosts
**System Containers (SC)**

- SCs are just Linux processes that present a fresh Linux machine (like a VM)!
- Connect SCs to logical LXC networks so they can communicate! Works across physical hosts via VXVLAN (Ethernet in L4 UDP)
- Restrict the resources (CPU, memory, disk) allocated to each SC (enforced by base OS via cgroups)



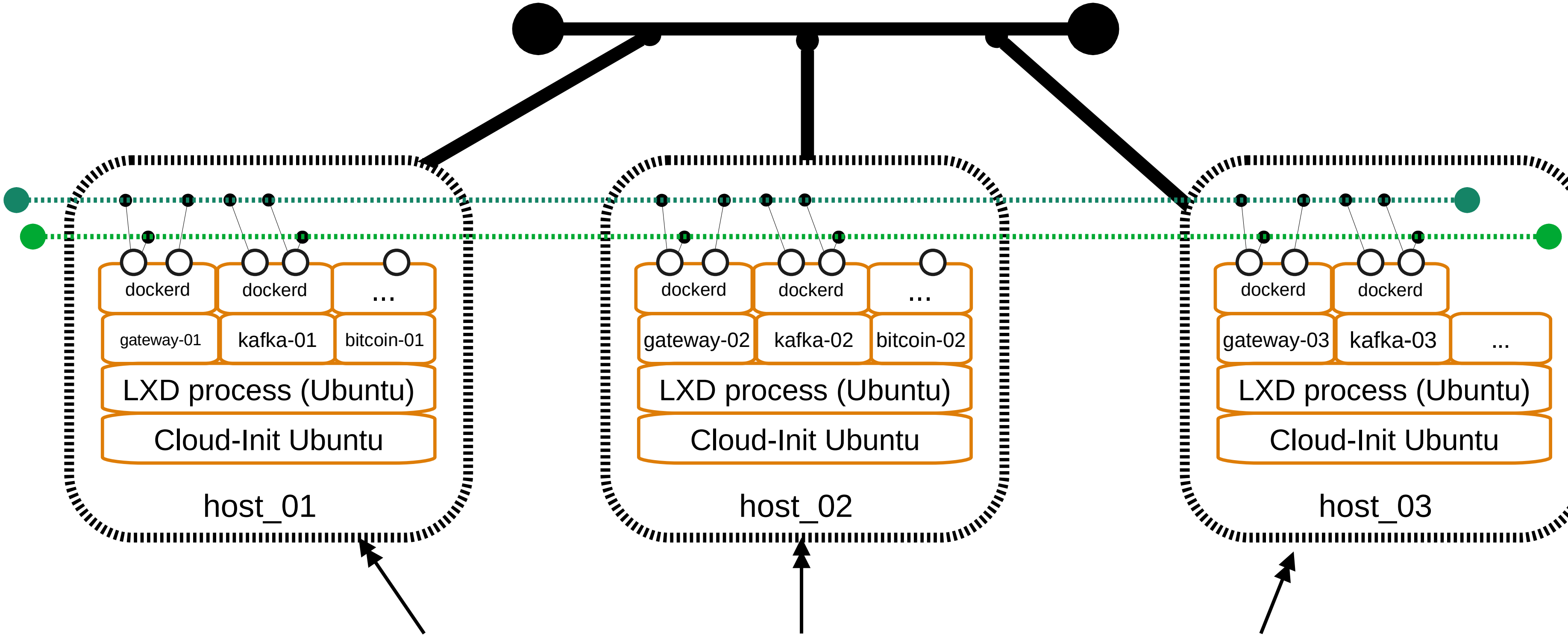
AC communicate via **docker swarm overlay network**. Traffic is encrypted when communicating across hosts!

- **App Containers** (e.g., bitcoind, TOR SOCKS5 proxy, lightningd, zookeeper & kafka brokers & logging & stream processing, etc.)
- Docker daemon process confined to each SC (1 dockerd per **SC**). Provides app-level container isolation within each SC.

Direct IP communication with other cluster members (no L3 NAT).

# Horizontal scaling –
# add commodity x86_64

Ethernet LAN WITH DHCP/DNS OR multipass private network (e.g., mpqemubr0)

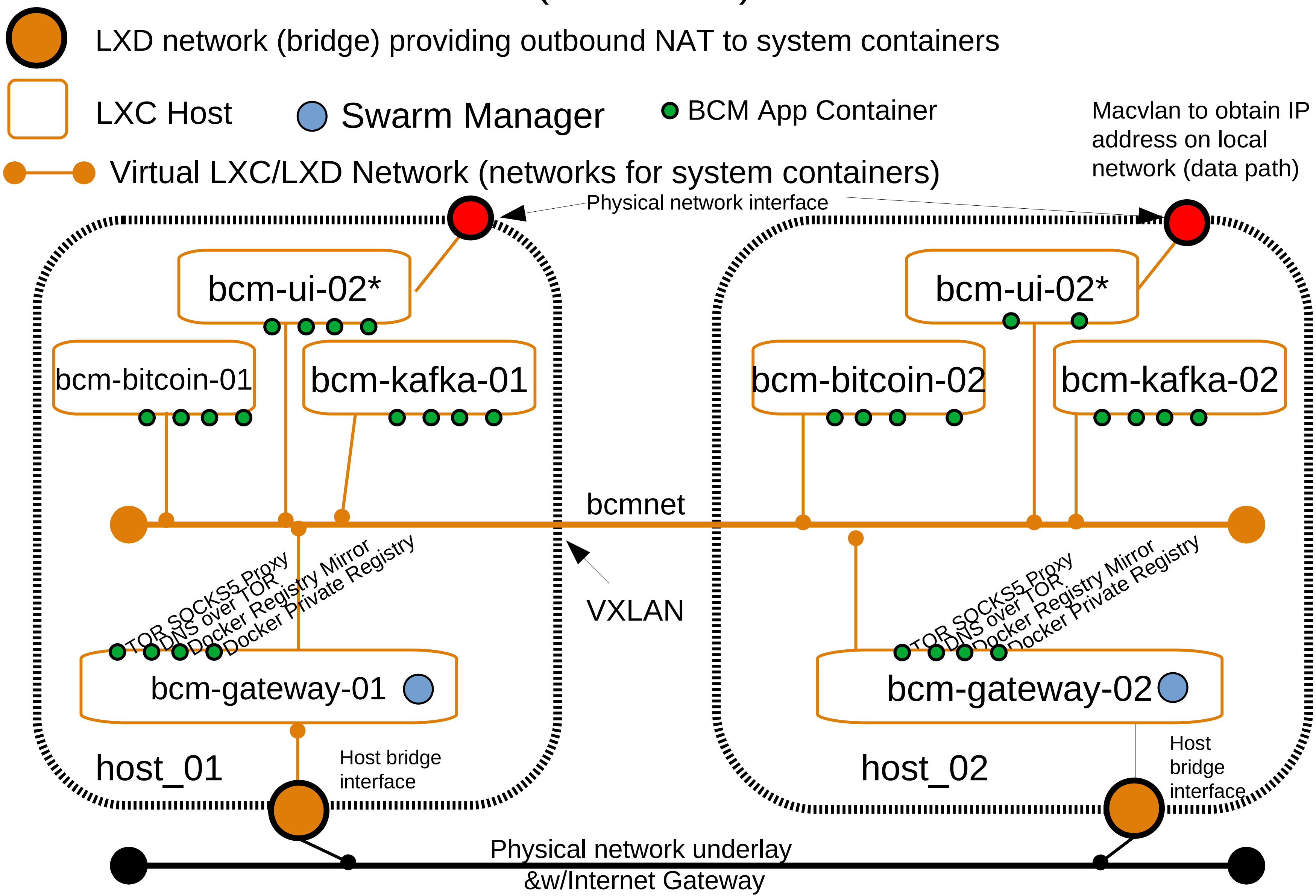


Physical Host with bare-metal Ubuntu OR ‘multipass’ VM provisioned with cloud-init

*Adding endpoints incrementally post-cluster creation is not yet supported.

# BCM System Container Network Topology

## (BCM Tiers)



● LXD network (bridge) providing outbound NAT to system containers

□ LXC Host

● Swarm Manager

● BCM App Container

●—● Virtual LXC/LXD Network (networks for system containers)

Macvlan to obtain IP address on local network (data path)

Physical network interface

bcm-ui-02*

bcm-bitcoin-01

bcm-kafka-01

bcm-ui-02*

bcm-bitcoin-02

bcm-kafka-02

bcmnet

VXLAN

TOR SOCKS5 Proxy
DNS over TOR
Docker Registry Mirror
Docker Private Registry

bcm-gateway-01

TOR SOCKS5 Proxy
DNS over TOR
Docker Registry Mirror
Docker Private Registry

bcm-gateway-02

host_01

Host bridge interface

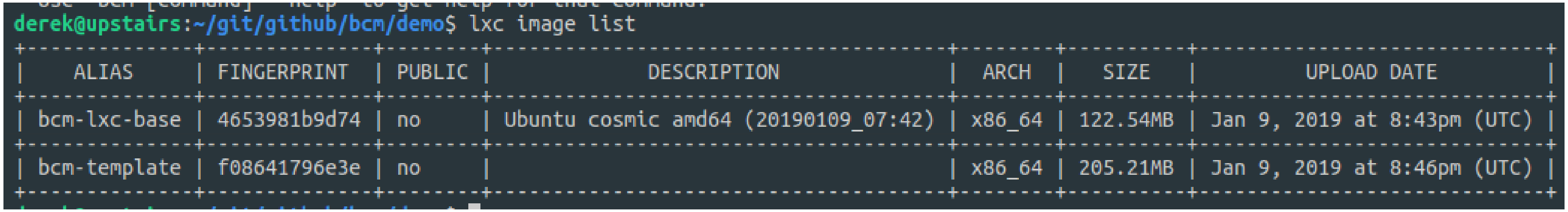
host_02

Host bridge interface

Physical network underlay
&w/Internet Gateway

# BCM host_template

- All system containers are cloud-based Ubuntu image from Canonical (currently cosmic=18.10)



```
derek@upstairs:~/git/github/bcm/demo$ lxc image list
+--------------+--------------+--------+--------------------------------------+--------+----------+------------------------------+
|    ALIAS     | FINGERPRINT  | PUBLIC |             DESCRIPTION              |  ARCH  |   SIZE   |         UPLOAD DATE          |
+--------------+--------------+--------+--------------------------------------+--------+----------+------------------------------+
| bcm-lxc-base | 4653981b9d74 | no     | Ubuntu cosmic amd64 (20190109_07:42) | x86_64 | 122.54MB | Jan 9, 2019 at 8:43pm (UTC) |
+--------------+--------------+--------+--------------------------------------+--------+----------+------------------------------+
| bcm-template | f08641796e3e | no     |                                      | x86_64 | 205.21MB | Jan 9, 2019 at 8:46pm (UTC) |
+--------------+--------------+--------+--------------------------------------+--------+----------+------------------------------+
```

- ‘bcm-template’ includes dockerd installation, tools for development, configuration files, etc.
- System containers are back by BTRFS volumes.
  - BTRFS provides RAID-like functionality, snapshots, Copy-on-Write
  - Host filesystem can be native BTRFS or you can run BCM on top of an EXT4 or LUKS encrypted volume

# BCM Tiers vs Stacks

- BCM Tier: collection of System Containers distributed across LXD Endpoints
  - Different deployment models: completely distributed, primary/backup, etc. Can impose rules (min/max hosts)
- `BCM Stack:` BCM app containers deployed across BCM Tier hosts (SC) using Docker Swarm
  - Docker swarm manages a cluster of docker engines
  - Declarative, scaling, desired state reconciliation, multi-host networking, service discovery, load balancing, rolling updates, etc.

*not implemented

# Gateway on BCM

Gateway Tier

The first five LXC hosts act as docker swarm managers bcm-gateway-xx

| **BCM App-Container** | • Port | **Purpose** |
|---|---|---|
| Tor SOCKS5 Proxy | • TCP 9050 | Outbound TOR for all SOCKS5-capable traffic. |
| TOR-enabled DNS | • TCP 9053 | DNS Queries over TOR. |
| Docker Registry Mirror | • TCP 5000/5001 | Public container cache (outbound TOR). |
| Docker Private Registry | • TCP 5010/5011 | Stores & serves custom build docker containers to downstream docker daemons. |
| Privoxy*/Squid | TBD | HTTP/HTTPS caching proxy (outbound TOR). |

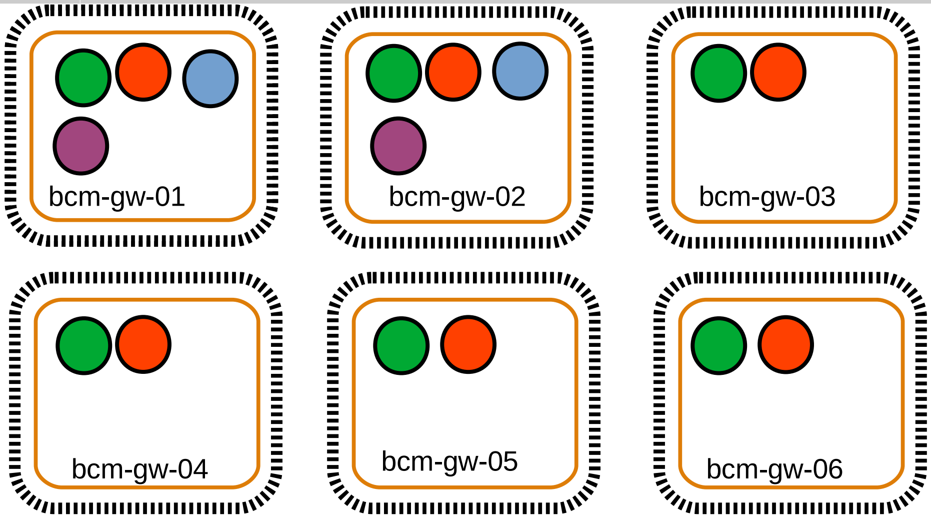
🟢 Tor SOCKS5 Proxy

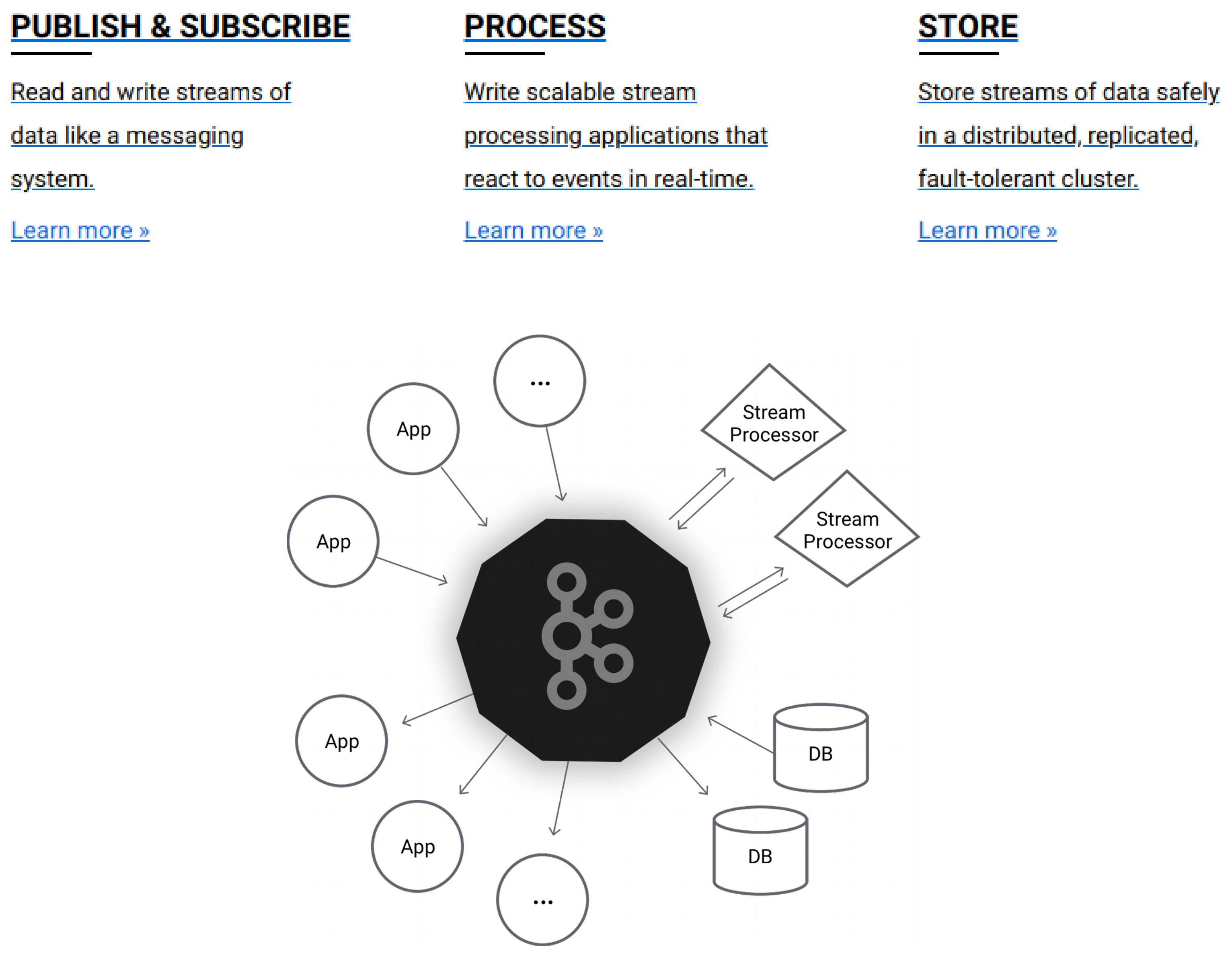
🟠 TOR-enabled DNS

🔵 Docker Registry Mirror

🟣 Docker Private Registry

bcm-gateway-01





## PUBLISH & SUBSCRIBE

Read and write streams of data like a messaging system.

Learn more »

## PROCESS

Write scalable stream processing applications that react to events in real-time.

Learn more »

## STORE

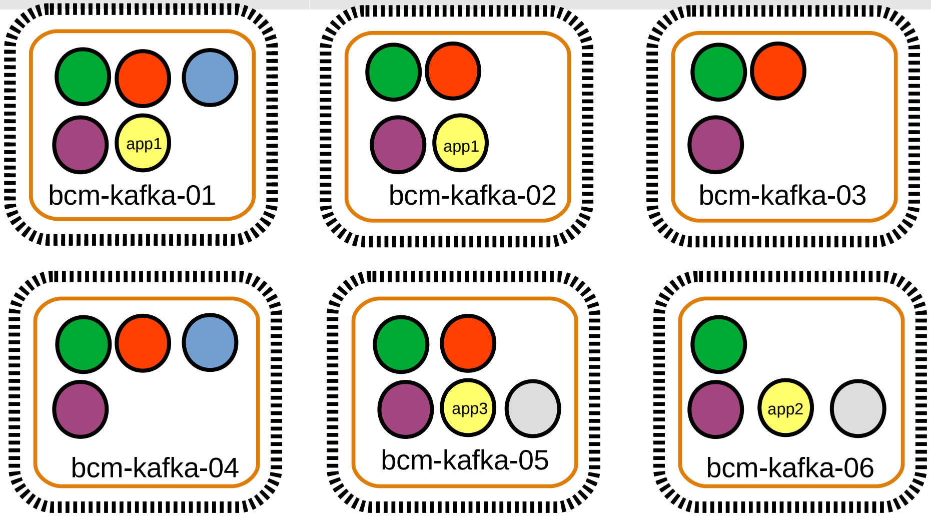
Store streams of data safely in a distributed, replicated, fault-tolerant cluster.

Learn more »

# Kafka on BCM

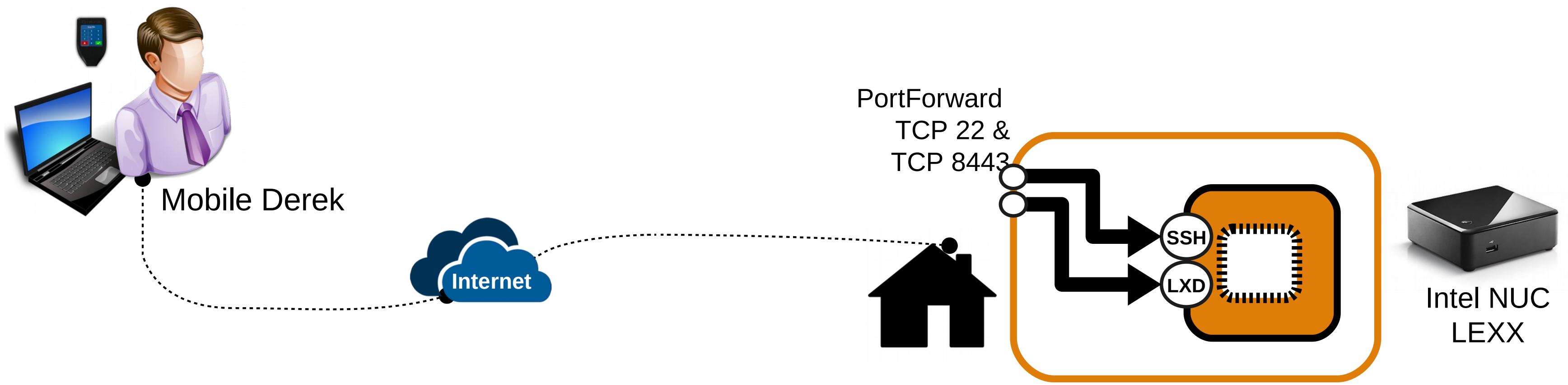
| BCM App-Container | • Purpose | Deployment Rules |
|---|---|---|
| Zookeeper | • Distributed Coordination | First 5 LXD endpoints. |
| Kafka Broker | • Distributed Messaging | EVERY LXD endpoint. |
| Kafka Schema Registry | • Stateless information store | Swarm managed, AT LEAST 2 instances |
| Kafka REST | • Stateless API for Kafka | Swarm managed, AT LEAST 2 instances |
| Kafka Connect | Stateless data input/output shipping | EVERY LXD endpoint (pairs with brokers). |
| Kafka Streams | Distributed coordination | Deployed by external docker swarm apps, scheduled on bcm-kafka-xx hosts. |

- 🟢 Kafka Broker
- 🟠 Zookeeper
- 🔵 Kafka Schema Registry
- 🟣 Kafka Connect
- ⚪ Kafka REST
- 🟡 App-specific Kafka Streams



# Demo Concept

- There’s an Intel NUC named LEXX with Ubuntu 18.04 installed & password-based open-ssh server.
  - Lexx is resolveable by the SDN controller.
- SSH service is publicly available over Internet via port forward*
- Note! All management plane activities are PLANNED to occur over TOR (or local network).



□ Exclusive Ownership & control boundary

⬚ Independent commodity x86_64

■ BCM Cluster

**Goal**: Bring LEXX under BCM management & deploy a new data center.
* Future work is to move all remote SSH commands to use TOR.

# Demo Explained

(1) Download BCM from github and run setup.

(2) Run **bcm init** to initialize your SDN controller.

(3) Create a BCM cluster by running **bcm cluster create**.

(4) Create a BCM Project with **bcm project create**.

(5) Deploy a BCM Project to a BCM Cluster using **bcm project deploy**.

```
# This script is written for demo purposes.
source "$BCM_GIT_DIR/.env"
source ./meetup.env

# run bcm init
bcm init --name="BCM" \
           --username="$BCM_CERT_USERNAME" \
           --hostname="$BCM_CERT_HOSTNAME"

# Create a basic project definition.
bcm project create --project-name="$BCM_PROJECT_NAME"

# new cluster based on existing SSH endpoint.
bcm cluster create --cluster-name=meetup --provider=ssh --hostname=lexx

# then deploy that project definition to an existing cluster.
bcm project deploy \
           --project-name="$BCM_PROJECT_NAME" \
           --cluster-name="$BCM_CLUSTER_NAME" \
           --user-name="$BCM_PROJECT_USERNAME"
```

# BCM Planned Services

- Cwtch for untrusted asynchronous communications.
- Nextcloud/Mastodon/Fediverse
- Bitcoin/Lightning/Sidechains,
  web-based wallets,
- Server-side infrastructure: block explorers, blockchain indexers, wallet servers (e.g., electrum), etc.
- Databases (Elastic) and Visualization (Grafana, Kinbana, etc) for data exiting the kafka processing pipeline.
- Bitcoin & Lightning Apps: lightning-charge, BTCPay Server,

# Planned Improvements

- Git auto-pull via TOR on BCM Controller
- GPG signing of ~/.bcm git repos using Trezor for authorization.
  - Use signed git tags as authorization for CI stack
- Operational
  - Trezor-based key-rotation.
  - Vulnerability assessments, port scanning, etc., as part of BCM development CI process.
  - Automate backup/restoration of Kafka topics
  - Georedundant backups using TAHOE-LAFS.
  - Protect data-at-rest by inserting FUSE layer*.
- Controller client side logging.

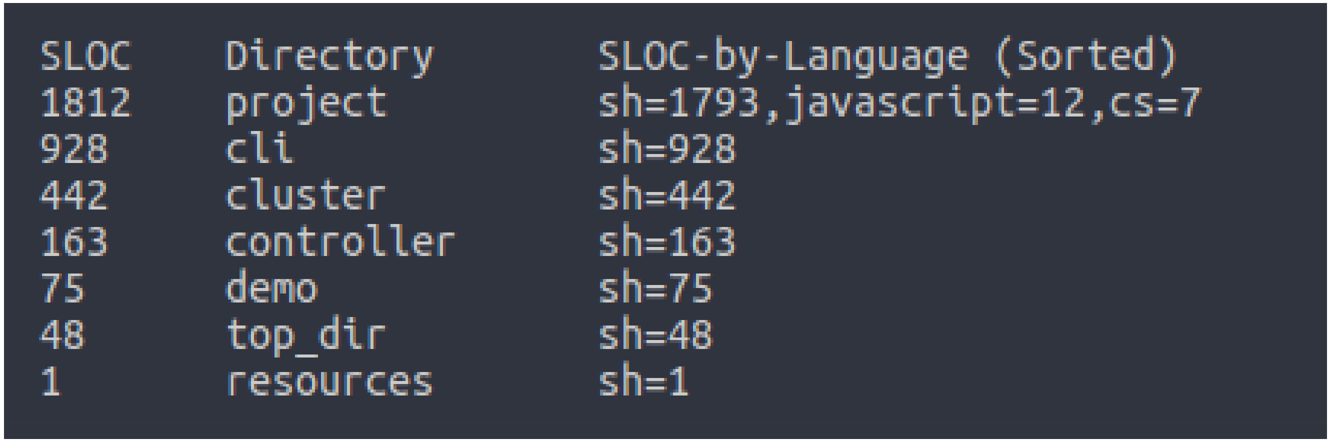
*You can always run BCM on top of a LUKS partition (full disk encryption).

# BCM Flaws and Weaknesses

- Technical:
  - Currently uses privileged LXC containers
    - Mitigated though dockerd process containment and the use of onion services.
  - No CI pipeline currently.
- Procedural
  - No development guidelines, format code review process, etc.

# BCM Stats as of 10Jan19

ALL BCM code development occurs in Visual Studio Code.
MOST of the codebase has been linted by ‘shellcheck’.



```
SLOC    Directory       SLOC-by-Language (Sorted)
1812    project         sh=1793,javascript=12,cs=7
928     cli             sh=928
442     cluster         sh=442
163     controller      sh=163
75      demo            sh=75
48      top_dir         sh=48
1       resources       sh=1
```

- Donations (BTC only):

3KNX4GTmXETtnFWFXvFqXg9sDJCbLvD8Zf