1) High-level design and decisions (what to choose and why)

1. Purpose & scope — decide what you want to learn:
    - Low-interaction (e.g., port responders) vs high-interaction (full SSH shell) — low interaction is safer and cheaper; high interaction gives richer data (real attacker activity and payloads).
    - Single honeypot vs honeynet (many sensors over regions) — start single, then scale.
2. AWS building blocks you'll use:
    - EC2 (honeypot instances), Security Groups, VPCs/subnets (isolation), IAM role for logging, S3 for storing logs, VPC Flow Logs, CloudWatch, and optionally GuardDuty for cross-analysis/alerting. AWS guidance recommends VPC security best practices and using GuardDuty and flow logs as part of monitoring. AWS Documentation+1
3. Choice of honeypot platform:
    - Cowrie — SSH/Telnet interactive honeypot, well-documented and lightweight (good single EC2 sensor). codygula.com
    - T-Pot — all-in-one platform (many honeypots + ELK) useful for single VM full stack. InfoSec Write-ups
    - MHN (Modern Honey Network) — server + distributed sensors to centrally manage many honeypots; good when scaling. GitHub

---

2) Architecture (recommended for production-ish testing)

- Create a dedicated AWS account or separate VPC (do NOT put honeypots in production VPC).
- VPC with public subnet for honeypot instances (internet reachable) and a separate management subnet (private) for logging/management servers (ELK / MHN server).
- Security Group for honeypot:
    - Inbound: open specific service ports you want to attract (e.g., TCP 22 and TCP 80) to 0.0.0.0/0 **only** for honeypot ports.
    - Inbound: admin SSH port (e.g., 2222) only from your IP.
    - Outbound: restrict where possible — but note some honeypots require outbound access to fetch updates; if you want to prevent pivoting, block outbound except to necessary services.

- VPC Flow Logs → S3 (and optionally Athena) for network forensic queries.
- Central logging: send honeypot logs to S3 and also forward to ELK/Opensearch or to MHN/T-Pot UI.
- Enable GuardDuty and CloudTrail for cross-correlation/alerts. aws.github.io+1

---

3) Concrete step-by-step: deploy Cowrie on AWS EC2 (single honeypot)

This is a compact, tested pattern — Cowrie on Ubuntu EC2, logs to S3 + optional ELK.

Preliminaries (assumes AWS console + CLI available)

- Create/identify an AWS account and region.
- Create an SSH key pair in the region (or import one).
- Note your admin public IP (for locking admin access).

Step A — Create VPC & subnets (isolation)

1. In AWS Console → VPC → Create VPC (CIDR: 10.10.0.0/16).
2. Create a public subnet (10.10.1.0/24) for honeypot.
3. Create an internet gateway and attach it.
4. Create a route table for the public subnet with 0.0.0.0/0 → IGW.

(You can also use the default VPC but dedicated VPC is safer.)

Step B — Create Security Group

- Name: honeypot-sg
- Inbound rules:
    - SSH for admin: TCP 2222 — Source: your-ip/32
    - Cowrie (fake SSH): TCP 22 — Source: 0.0.0.0/0 (if you want to attract internet scanners)
    - HTTP (if running web honeypot): TCP 80 — 0.0.0.0/0
- Outbound rules:
    - Deny all by default, then allow minimal egress (e.g., DNS/HTTP) if you need updates. Consider using a NAT for controlled egress to a proxy.

Important: Ensure your real admin SSH listens on a non-default port (2222) and is restricted to your IP. Cowrie will bind to port 22 for the fake shell. Several tutorials recommend this pattern. codygula.com

Step C — Launch EC2 (Ubuntu 22.04 recommended)

- Instance type: t3.micro or t3.small (for low traffic). Choose larger if expecting heavy activity.
- AMI: Ubuntu Server 22.04 LTS.
- Subnet: public subnet created earlier.
- Assign Public IP: yes.
- Security group: honeypot-sg.
- Key pair: your key.

Step D — Basic hardening for management

SSH to instance using admin port (example if you open port 2222 for admin):

```
# from your workstation (adjust username/ec2-user accordingly)

ssh -i mykey.pem -p 2222 ubuntu@<EC2_PUBLIC_IP>
```

On the instance:

```
# update & install essentials

sudo apt update && sudo apt upgrade -y

sudo apt install -y git python3-venv python3-pip build-essential libssl-dev libffi-dev

# create a non-root user for cowrie if needed
```

Step E — Install Cowrie (quick install)

This example follows the common approach used in several guides — configure a virtualenv and run Cowrie as a service.

```
# create directory

sudo adduser --disabled-password --gecos "" cowrie

sudo su - cowrie

git clone https://github.com/cowrie/cowrie.git

cd cowrie

python3 -m venv cowrie-env
```

```
source cowrie-env/bin/activate

pip install --upgrade pip

pip install -r requirements.txt

# copy template config and edit

cp etc/cowrie.cfg.dist etc/cowrie.cfg

# edit etc/cowrie.cfg - set hostname, enabled protocols, log targets (JSON)

# Example: configure journald/file output and enable JSON logging for downstream parsing
```

Create a systemd service (run as cowrie user) so it restarts on failure. Many guides show exact systemd unit files — follow Cowrie docs.

(Several step-by-step walk-throughs exist for exactly these steps.) codygula.com

Step F — Send logs to S3 & ELK

- Cowrie logs to var/log/cowrie and supports JSON output.
- Use awscli or a lightweight Filebeat agent to ship logs to:
  - S3 (for long-term storage and forensic retrieval).
  - Elastic/OpenSearch (for search/visualization).
    Example to copy logs to S3 (cron or logrotate hook):

```
# install awscli and configure minimal IAM role on EC2 instead of local creds

sudo apt install awscli

# upload

aws s3 cp /home/cowrie/cowrie/log/ s3://my-honeypot-logs/cowrie/ --recursive
```

Better: attach an instance IAM role with s3:PutObject and use a small agent to push logs.

Step G — Enable VPC Flow Logs & GuardDuty

- In VPC console → Flow Logs → Create flow log for the honeypot subnet; destination S3 or CloudWatch Logs.
- Enable GuardDuty in the account and region — it will pick up suspicious behavior and integrate with CloudWatch events for alerting. aws.github.io+1

Step H — Verify & test

- Check Cowrie UI/console logs.

- Run a simple external scan (from a separate test machine) against port 22 and see if it records the login attempts and commands.
- Validate logs in S3 / ELK.

---

4) Scale / alternatives: T-Pot and MHN

- T-Pot: all-in-one, runs many honeypots + ELK Dashboards on one host (typically runs as an appliance image). Good for rapid deployment; many guides for EC2 exist. Use a larger instance (m5 / c5) for T-Pot. InfoSec Write-ups+1
- MHN: deploy MHN server (one EC2) and many lightweight sensors (EC2 or on-prem) that register with the MHN server. MHN provides central deployment & data collection. Good for distributed honeypot fleets. See the MHN GitHub for sensor types and instructions. GitHub

---

5) Logging, analysis, and detection workflows

1. Collection:
   - Cowrie/T-Pot/MHN → JSON logs to S3 and to a central ELK/OpenSearch.
   - VPC Flow Logs → S3/CloudWatch → Athena queries for IP patterns.
   - CloudTrail & GuardDuty → CloudWatch Events (for correlation).
2. Processing:
   - Filebeat/Logstash → parse Cowrie JSON → index in Elasticsearch/OpenSearch.
   - Enrich logs with GeoIP, ASN, and known bad IP lists.
3. Analysis you should run regularly:
   - Top attacking IPs (count attempts).
   - Brute-force username/password pairs and common payloads.
   - Timeline of actions per IP (to detect multi-stage intrusions).
   - File artifacts (downloaded binaries) — quarantine and run in sandbox (Cuckoo / detonation lab).
   - Correlate with GuardDuty findings and CloudTrail to spot if any real resources were accessed.
4. Alerts:

  - High volume of failed logins from same IP → alert.
  - Successful session with file download → high severity (exfil/payload).
  - Known C2 indicators observed in outgoing traffic from honeypot → alert and isolate.

(Plenty of guides show how to wire this into ELK or MHN dashboards.) GitHub+1

---

6) Data analysis example (what insights you can get)

- Attack surface profiling: which ports/protocols are most frequently probed, from which countries and ASNs.
- Attack behavior: common commands run (recon commands, privilege escalation attempts), scripts uploaded, file names used for malware.
- Malware capture: binaries and their metadata (hashes) for IOC creation and sandboxing.
- Trend analysis: time/day patterns, correlation with known exploit disclosures (e.g., new CVE scanning spikes).

Use Athena on S3 flow logs or Kibana dashboards to quickly answer such questions.

---

7) Risks, containment, and legal/ethical notes

1. **Pivot / abuse risk:** a fully interactive honeypot can be used by attackers to launch attacks. To reduce risk:
   - Strongly restrict outbound traffic from honeypot (prevent direct internet access except to controlled update hosts).
   - Monitor and block any attempt to escalate or access AWS metadata endpoints (IMDS). Block IMDS access from honeypot. (IMDS is a common AWS pivot target.)
   - Use network ACLs and routing rules to isolate the honeypot.
2. **Legal & policy:** running a honeypot that captures personal data or allows law-breaking activity may have legal implications. Check your organization's policy and, if needed, legal counsel.
3. **Do NOT connect honeypots to production networks** — always isolate. Multiple tutorials and AWS posts stress isolating honeypots and using monitoring services. AWS Documentation+1

---

8) Cost & cleanup

- Cost drivers: EC2 instance hours (especially bigger types for T-Pot/MHN), S3 storage for logs, data transfer if you ship logs out of region, EBS volumes, and any ELK/OpenSearch provisioning.
- Test with t3.micro first; scale up for heavier logging or T-Pot.
- Cleanup: terminate EC2, delete EBS snapshots, remove S3 objects, disable VPC Flow Logs, and remove IAM roles/policies.

---

9) Quick checklist / runbook (copyable)

- Create separate AWS account or VPC for honeypots.
- Create VPC, public subnet, internet gateway.
- Create security groups (admin restricted, honeypot ports open).
- Launch EC2 (Ubuntu), attach IAM role for S3 PutObject and CloudWatch.
- Install Cowrie (or T-Pot/MHN), configure to output JSON.
- Configure VPC Flow Logs → S3.
- Configure CloudWatch / GuardDuty / CloudTrail.
- Configure log forwarding to ELK/OpenSearch and set dashboards.
- Set automated snapshot & retention policy for S3 logs.
- Test by running controlled scans and verify logs & alerts.
- Hardening: block IMDS for the honeypot (e.g., iptables or VPC route policies), restrict outbound traffic.
- Document cleanup steps & cost thresholds.

---

10) Resources & further reading (selected)

- AWS VPC security best practices (VPC Flow Logs + GuardDuty recommendations). AWS Documentation
- Cowrie SSH honeypot install guides (practical walkthroughs). codygula.com
- Modern Honey Network (MHN) GitHub (centralized management for many sensors). GitHub

- T-Pot / all-in-one honeypot platform guides and deployments. InfoSec Write-ups
- AWS Security blog posts about deception/honeypot approaches on AWS. Amazon Web Services, Inc.