A.R.E.S. Phase One Execution Kit

1. Deployment Blueprint

Coreframe Container Layout (Docker Compose)

```
version: '3.9'
services:
  caddy:
    image: caddy:2
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./Caddyfile:/etc/caddy/Caddyfile
      - caddy_data:/data
    networks:
      - internal
    restart: unless-stopped

  ares:
    build: ./ares
    depends_on:
      - memory
    networks:
      - internal
    environment:
      - MEMORY_URI=memory:6379
    deploy:
      resources:
        limits:
          cpus: "2"
          memory: 2G
    restart: unless-stopped
    user: nonroot

  memory:
    image: redis:7
    volumes:
      - memory_data:/data
    networks:
      - internal
    restart: unless-stopped

  observer:
    build: ./observer
    networks:
      - internal
    restart: unless-stopped

volumes:
  caddy_data:
  memory_data:

networks:
  internal:
    driver: bridge
```

2. Caddy Reverse Proxy Configuration

```
coreframe.example.com {
    encode gzip
    tls {
        protocols tls1.3
    }
    @whitelist {
        remote_ip 192.168.0.0/16
        remote_ip 203.0.113.0/24
    }
    respond @whitelist 403
    reverse_proxy /api/* ares:8000
    log {
        output file /var/log/caddy/access.log
        format json
    }
    rate_limit {
        zone default {
            events 10
            window 1s
        }
    }
}
```

3. Observer Protocol – Live Dashboard Setup

Tools: cAdvisor, Grafana, Filebeat, ElasticSearch, Kibana

Command to run cAdvisor:

```
docker run   --volume=/:/rootfs:ro   --volume=/var/run:/var/run:ro   --volume=/sys:/sys:ro   --volume=/var/lib/
```

4. Inner Circle Tester Briefing

Mission Brief, Access Instructions, Engagement Rules

SSH Access:

```
ssh -i ./keyfile innercircle@coreframe.example.com -p 2222
```

API Access:

```
curl -X POST https://coreframe.example.com/api/v1/command   -H 'Authorization: Bearer <token>'   -H 'Content-Ty
```

5. Engagement Cycle Script

```
#!/bin/bash
# PHASE ONE ENGAGEMENT CYCLE
DATE=$(date +"%Y%m%d_%H%M%S")
HASH=$(sha256sum /data/memory_stream | awk '{print $1}')

echo "=== SNAPSHOT START ==="
echo "Time: $DATE"
echo "Hash: $HASH"

# Sanitize tester data
jq 'del(.tester_personal_info)' /data/memory_stream > /data/memory_stream_sanitized
```

```
# Rollback containers to known good state
docker compose down
docker compose up -d

# Archive logs
tar -czf archive_logs_$DATE.tar.gz /logs/observer
echo "=== SNAPSHOT COMPLETE ==="
```

6. Threat Playbook Auto-Capture

```
def record_codex_entry(pattern, reasoning, outcome):
    entry = {
        "pattern": pattern,
        "reasoning": reasoning,
        "outcome": outcome,
        "timestamp": datetime.utcnow().isoformat()
    }
    with open("/codex/codex_patterns.json", "a") as f:
        json.dump(entry, f)
        f.write("\n")
```

7. Launch Procedure

1. Environment Prep: Pull latest container images, verify firewall rules & VPN access. 2. Access Distribution: Generate per-tester SSH keys + API tokens. 3. Dry Run: Simulate prompt injection & lateral movement internally. 4. Go Live: Notify Inner Circle of engagement window, activate monitoring dashboard. 5. Cycle Close: Run engagement script, archive logs + Codex updates, conduct debrief.