Sandbox

VPN Cloud

Provide MVP (minimal viable product).
Define product: we are creating a product to solve inter-connectivity.
Flesh out a product with sufficient features to satisfy early adopters.

Additional features are only developed after considering user feedback from initial MVP rollout.

Define MVP:

- Provide user with low tech, easy plug and play configuration
- Easily connect to 44net via set of methods such as Wireguard
- Access Hamnet and ARDEN, and other Ham networks
- Provide publicly routable IP in 44 prefix
- Provide download of VPN software on site
- Allow users to either request more than 1 IP or ability to use own 44net prefix range
  If latter add BGP peering.
- Display tunnel status (connectivity,traffic)
- A user side / Developer api should also be created to foster experimentation with VPN platform.

Design Responsibly:

- Avoid over building (avoid bloat ware), build to scale
- Create a feature list:
  - Must have / critical for functionality
  - Could be useful
  - Like to have
  - Neat but not needed
- Don't build out all features before deployment, aim for functional product adding features as time allows
- Build out easy to implement features first, then adding more difficult to develop.

Define personnel and duties (initial):

- 2 programmers, frontend/backend full stack developers [full time].
- 1 Network engineer (NOC) [PTE / or on retainer]
- 1 Technical Supervisor [full time] – As coordinator
- Help desk / Junior Network Technician [PTE or volunteers]

End user interaction considerations:

- Backend Infrastructure code and setup should not be made available as publicly viewable repository.
- The front end repository SHOULD be made available to users, along with a token based API that gives users the ability to experiment with developing their own tools and interfaces.
- The front-end repo should also accept user branch enhancements that can be merged with production workflows.

Financial:

- Will Require budget, and Ability to Hire.
- Salary demands vs volunteer commitments.
  - Cannot expect forward momentum on project without a few stake holders and minimally paid PTE experts in various fields.
- Provide budgetary plan and user engagement goals.
- If providing hardware or software, define key suppliers, note any backup suppliers if logistics issues.
- Define ongoing revenue streams (how will service be funded)
- Define costs major expenses (for next year)
- Detail project milestones.
- Are there any outside funding sources that can augment ARDC, ARRL, external grants.
- Add minimum of 20% to end line budget for contingencies.
- Define critical risks and assumptions for venture.
- What strategies are planned to overcome these risks?

Designing:

Writeup tech spec or feature
Design low-fidelity mockups.
Do feed back rounds
Design components based on feed back.

Feature planning:

Writedown problem to be solved
Add task priority to each feature

Product launch:

Website for beta testers
Social media teasers
Beta phase blog on own site
Post to blogs, media release

Marketing:

- Foster relations with other vendor outlets to create synergetic products
  (new distribution channels, i.e. gl-net including ARDC connectivity entry in router)
- Via marketing channels promote product.
- Develop strategy to get organic traffic to service
- Build email subscriber list

Design consideration's

Scalability

- After prototype built, may want to determine
  if additional capacity should be done as a distributed infra with Ansible VM launch , or move to a master / worker scheme.
- Static data provided by CDN
- Dynamic content : stats, changing info use Micro-caching

Security
Persistence

- Stress test system using automation such as Ansible or terraform

Fault tolerant

- Nginx for front end reverse proxy and/or Haproxy
  - nginx can handle >10k concurrent connections

Updates/patching
Connectivity

Initial Infrastructure

Setting the Stage: (start small and scale)

Core POPs:

- 1 - 2 continental core POP's (USA, EU)
- ASR1001 core Router w/ min 1gig connection to at least 2 BGP peering for announcing Prefixes.(44net IP's)

Edge POP's:

- Add edge POP's as tunnel endpoints proportionally to the increase of user activity.
- Each 2vCPU edge POP to provide tunnel endpoints for up to 250 users (250 tunnels is anecdotal limit for Wireguard connectivity- we will be testing this assumption with live users)

Servers (can be virtual machines)

RR (route reflector) - Peer with Core router. Route reflector helps rangel the (N* (N-1) / 2) iBGP mesh connections.

BGP -Users peer with BIRD on this machine using auto configure platform.
**Each user has option to get a free Private ASN or bring their own Public ASN.

Separate VM or docker containers for each network were meshing. IE Hamnet link server.

(2) User facing Website. (test/staging, production)
(2) Admin facing Website (test/staging, production)
(2) Backend portal (where the magic happens)
Database (use mongodb. can use hosted shards from DBaas.)

Development machines (many) —

- Creation done on developers own machines, then with pull request, using git workflows, auto merging uploaded to staging for live testing.

Use multi stage development.(develop, staging/test, production)

- Each developer to make changes to own sandbox Machines before making pull request to merge into testing branch. have a fully functional (live) testing/staging platform where development API calls are tested against.

Use Github workflow:

Pull request of staging branch, commit to workflow.
Test staging platform.
If successful, request pull of production branch.

GITHUB COORDINATOR:

- Compare pull branches/merge requests
- Coordinator can be a programmer.
- Verifies branch request change's before merging into staging
- development branch and merges from staging to production.
- Setup continuous integration using github actions
- Delete merged branches as needed.

Q & A / documentation on Site:

- Have users post common questions to a mailing list or discord stream.
- Crowd sourcing answers will relieve many of the non critical issues from ticketing system.
- Periodically scan mailing list and discord channels to scrap question/answer combos and add to a knowledge base on webpage.
  - How To's
  - How to setup your own point to point connection
  - How to utilize different products.

Code Policy:

- Define code language to be used: Javascript, Python, NodeJS etc.
- For python portion of code, use threading (ThreadPoolExecutor)
- Use synchronous code when possible on python and NodeJS , use await when function results need to wait for a return.
- Be transaction aware so a delete in one location is acknowledge in another. don't allow a deletion in DB for user account leave a tunnel active.
- For redis, use optimistic locking with the .watch command
  - https://realpython.com/python-redis/#getting-started

A few Features to include:

- As user is signed up, assign a support person to help with setup using ticketing software like os ticket
- Use an api to allocate volunteers todos
- Provide toast on web ui as database or functions that are time intensive progress
- Use queuing system for email functions and long run database function
- Use redis/memcached for peer stats
- DB stores current stats for all users
- Loop db based on public key for specific users stats
- Create random tunnel name.
- Net mapping a single public ip and port with a private ip with a
- Add ipv6 /48 block
- Add ipv4 /29. or /27
- As IP's are used add to db
- As wg0 added or deleted update
- Modify prototype; move python portal to separate machine with Wireguard cli control through secure connection.
- Confirm email before being able to login
- Ansible setup pi
- Get next ip based on ip that doesn't have peer public key
- Port forward attached to DNS
- Send user copy of config via email.
- User's ip gets registered with packetframe as domain name.
- Add enabled user, so admin and user can disable tunnel
- Add max tunnels using a max in user profile
- Redis for user stats, handshake
- Server stats in mongo db
- Unified webpage interface (webpage consistent)
- Add pre-shared key as feature
- Add persistent keep-alive
- Add timeout to auto delete tunnel based on time
- Add date for temporary disable to db
- Add last handshake
- Add transfer in/out. bottom
- Add QRcode
    - QRcode and a downloadable config available on each tunnel
- Use geolocation of user's ip ip-api.com to determine closest regional pop to use
- Add api ip blocklist neutrino api
- Text via twillio
- Add port firewall to VPN tunnel via interface
- For /24 create a altDB entry for user use ARIX ipam as starting point
- Give each user a routable ipv6
- ASN number using a private range for those that don't have public ASN's
- Verify public ASN against peering-db email on file. peering-db has api. OR use regional IRR's own DB
- Rate limit access for getting new VPN (including add/delete)
- Ansible playbooks for spinning up vm
- Bird auto peer
- For DB add race condition verification
- provide a means to limit tunnels per user

- provide means to add ip's
- Provide Admin interface
- Provide notice of inactivity after 1 month
- Offer help with ticket link to volunteer
- Offer notice every 3 months there after.
- Provide auto trimming after 1 year with 30,14, 2,1 day pre-notice
- Notification's use publish/queuing
- Use task manager to limit wait time for users, i.e if it takes 1-2 seconds to validate user and forward to next function,, 1 to create config , 1 to update DB and 3 second to send email, its better to handoff to a task manager to improve user retention. A user's inclination to leave a site or give up on a process increases by 25% after 4-5 second delay.
- Provide user feedback with toast or ….
- Personal user content use Direct DB pull.
- Inter server messaging via published/ que's
- Use paramiko to control dumb end Pop's from back-end controller.
- Provide a stats page showing list of users and their url
- What are customer metrics.
- A RESTful api to be provided for administrator access using an api token.
- An automated configuration will be sent to users email upon tunnel creation.
-

Client Download:

- Page can be provided with each platform of software. using web agent's, the OS software that matches the users browser id can be presented to simplify installation.

Open sources software used:

pathvector (ham made)
prometheous
graphana
packetframe (ham made)
git lab
mongodb
wireguard
Bird / FRR bgp
Proxmox
Queing or Pub/sub with beanstalk or like

—**non cost software / services**

Ansible
Redis /memcached
HA proxy
Docker

—**paid software / services**

Twillio (SMS)

NTP: accurate time is needed for multi node synchronization

MONITORING:

- Prometheus with Graphana to visualize site and system statistics and loads.
  - Monitor:
    - server specs/resources (cpu, memory, hard drive, etc).
    - number of connected users.
    - backups.

LOGS:

- System logs are available.
- Only details that are logged are ip's, and errors.
- Send console and server errors to external log analyzer. DataDog or like.

SECURITY:

- Never send bare keys via email. use encryption
  - **Security Certificate:** LetsEncrypt is a free services that allows certificate signing automatically using cert bot. using a port 80 acme html domain validation process. The entire infrastructure can be given ssl certificates
  - **BGP Security** These are the most common BGP threats:
    - BGP routing table manipulation
    - BGP route spoofing
    - BGP DoS
      - add security such as: (plus standard configs)
        - neighbor 44.190.40.1 password 50M353CR3T
        - neighbor 44.190.40.1 ttl-security hops 1 address-family ipv4
        - neighbor 44.190.40.1 maximum-prefix 1000
    - Private tunnels for sensitive communication between machines
      - put machines on private vxlan

EXPERIMENTATION:

- Mpls movement of ips from one location to another.
  use mpls overlay between pops
  - IP's still will have to originate from the original datacenter
    but user will be able to connect with a local datacenter and use the higher capacity and higher speed pipe between datacenters.
  - The user chooses a new pop and the system creates a mpls link between locations.