Service Overview

ShinySpace maintains several services, such as this wiki. Find out more in this book.

- Short Overview
- ShinyDiffusion
- ShinyBookstackC
    - ShinyBookstack
    - Welcome to ShinyBookstack
- Forge: Forgejo (Git hosting, actions, etc)
- meet.shiny.space - Jitsi
    - Setup
    - Jitsi compose.yml
    - nginx reverse proxy configuration
    - Config - ENV and interface customization
    - STUN / TURN server

Short Overview

Short list that is hopefully easy to update. Running on shiny.space itself:

Public services:

forge.shiny.space - git repos, forgejo webui, including git lfs (currently no signups, but public repos possible)
meet.shiny.space - voice/video call with jitsi meet (because fuck discord)
wiki.shiny.space - this bookstack (public only viewable, no signups)
photos.shiny.space - public photo gallery, enjoy.
blog.shiny.space - new blog, replaced old blog with a simple solution using zola

files.shiny.space - DEAD - super simple file sharing with web UI. will be rebuilt someday

Invite only:

sftp - fileshare for collaboration
jellyfin.shiny.space - NEW - netflix-like application to play owned video files on the server from anywhere

Private:

ssh bastion host - ideally only for admin access, locked down vm and sshd
calDAV and cardDAV - sync contacts and calendars with own server
trilium.shiny.space - private synced notes/documentation, similar to obsidian but FOSS (but less features)
pics.shiny.space - automatic backup of photos from phone. replace google photos and similar.
api.shiny.space - NEW - http API endpoints for fun new shiny.space functionality.
grafana.shiny.space - NEW - insight into various stats, under development

Support services on other hosts:

vps.shiny.space:
ntfy - not selfhosted, but using scripts + app for easy alerts on mobile phone if shinyspace goes down.
coturn - STUN/TURN server running on a static, public IP for easy NAT Traversal - used for jitsi meet
iperf3 - test network connection speed (only enabled when needed)

funkraum.ch:
mail - running a mailserver because i didn't like any of the popular solutions. so far it works (2025)

ShinyDiffusion

ShinyDiffusion is a self-hosted instance of (currently) Automatic1111's Stable Diffusion Webui. It easily lets users create pictures from text prompts, some of which you see when looking at Shelves and Books on this website.

ShinyDiffusion currently only runs on demand - running a powerful PC with GPU 24/7 is a huge waste of power, since it only gets used once in a while. Ask the admin to run it ;)

ShinyBookstackC

This website.

ShinyBookstack

You're currently using this service.

Bookstack is an open source Wiki platform.

It is used to share and keep track of info about ShinySpace and all it's related topics, such as Game Development, Self-Hosting and more.

Welcome to ShinyBookstack

This website contains wiki-like information about ShinySpace, related projects, and entirely different projects that are cool.

Check out Shelves for an overview of topics.

Read ShinyBookstack for more info on this website.

Read more about how ShinySpace works: ShinySpace Tech Stack

Game Dev Projects:

RadGame by cheeseplease

HealthyGame by the shinyspace admin

Forge: Forgejo (Git hosting, actions, etc)

shinyspace runs a Forgejo instance. This is so we can easily work together with people who are used to e.g. GitHub workflows.

~~Forgejo has a dedicated VM: forgejo1~~
Forgejo now runs in a rootless podman container on brix4, making many things easier - no hard ram limits or storage limits for example, if needed it can use the entire server (unless other containers need resources too)

on it, there's a user "shinypod" that has podman permissions (subid/subgid stuff) and is configured to run Podman on (2024) ~~Debian 11~~ Arch Linux (because debian's podman version was quite old and didn't support some features needed for rootless) without root, meaning /etc/containers/containers.conf had to be adjusted to use cgroupfs.

the Forgejo pod was created with this compose.yml file:

```
version: '3'

networks:
  forgejo:
    external: false

services:
  forgejo:
    image: codeberg.org/forgejo/forgejo:8-rootless
    container_name: forgejo
    user: "1000:1000"  # Adjust this to match your host UID:GID
    environment:
      - FORGEJO__database__DB_TYPE=postgres
      - FORGEJO__database__HOST=db:5432
      - FORGEJO__database__NAME=forgejo
      - FORGEJO__database__USER=forgejo
      - FORGEJO__database__PASSWD=Wt8ooyMTMyD4wSz47I
    restart: always
    networks:
      - forgejo
```

```
    volumes:
      - ./forgejo_data:/var/lib/gitea
      - ./forgejo_config:/etc/gitea
      - /etc/localtime:/etc/localtime:ro
    ports:
      - "3000:3000"
      - "2222:2222"  # Note: Changed from 222 to 2222 for rootless container
    depends_on:
      - db
    userns_mode: "keep-id"

  db:
    image: postgres:14
    container_name: forgejo_db
    restart: always
    environment:
      - POSTGRES_USER=forgejo
      - POSTGRES_PASSWORD=Wt8ooyMTMyD4wSz47I
      - POSTGRES_DB=forgejo
    networks:
      - forgejo
    volumes:
      - ./postgres_data:/var/lib/postgresql/data
    userns_mode: "keep-id"



x-podman:
  in_pod: false
```

which, if ran on a new system, should create the latest version of forgejo7 (such as 7.4). Persistent data is in the ./forgejo directory, that's why it is mounted.

Updates should be easy and smooth on forgejo 7. updating to 8 is likely going to require manual intervention.

Attention: podman won't restart containers by default. i made the following systemd unit in this case:

.config/systemd/user/compose-forgejo.service

```
[Unit]
Description=Podman Compose MyService
Wants=network-online.target
After=network-online.target

[Service]
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=%h/forgejo
ExecStart=/usr/bin/podman-compose up -d
ExecStop=/usr/bin/podman-compose down

[Install]
WantedBy=default.target
```

the Type and RemainAfterExit lines are what keeps the container up.

A forgejo runner was set up for the shiny.space organisation, so any repos belonging to that can use it. it runs on a dedicated VM runner1 with Docker (as the required container sockets come with docker per default, but require extra setup on podman).

it was created (after installing dependencies such as docker) with the command

```
./forgejo-runner register --no-interactive --token <TOKEN> --name runner --instance
https://forge.shiny.space --labels docker:docker://node:16-bullseye,self-hosted
```

and started as a systemd service running /root/forgejo-runner daemon

meet.shiny.space - Jitsi

Jitsi is an open source video conferencing software that fully supports hosting it yourself for free. It also optionally comes with some neat features like whiteboard(excalidraw) and shared document editing(etherpad).

Setup

Jitsi can be ran in docker/podman containers. currently it runs on brix4, with an external STUN/TURN server at vps.shiny.space for clients with NAT issues or strict firewalls.

The basic setup is described in the docs: https://jitsi.github.io/handbook/docs/devops-guide/devops-guide-docker/#quick-start
it boils down to: download release, unzip it, take the needed files (honestly it's only like 2 or 3... docker-compose.yml, gen-passwords.sh, env.example if needed), create the config directory where persistent data and custom config will be stored, set the desired ENV vars in .env, and run it with podman-compose/docker-compose/whatever.

At Shinyspace, we also needed some setup on the reverse proxy. find it here:
https://wiki.shiny.space/books/service-overview/page/nginx-reverse-proxy-configuration
Additionally: ports needed are 80/443 for web traffic, and 10000 for the jitsi video bridge - weirdly enough it worked in some situations without the bridge, but not always.

Here's the compose.yml, edited by lucas to actually forward the needed ENV vars in podman, it appears Docker handles env vars differently but podman-compose needs the ${ENV_VAR} part. it was manually added only where needed... fix later.

link: https://wiki.shiny.space/books/service-overview/page/jitsi-composeyml

it also has the systemd service set up for easy restarting: ~/.config/systemd/user/compose-jitsi.service

```
[Unit]
Description=Podman Compose MyService
Wants=network-online.target
After=network-online.target

[Service]
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=%h/jitsi
ExecStart=/usr/bin/podman-compose up -d
ExecStop=/usr/bin/podman-compose down
```

```
[Install]
WantedBy=default.target
```

Note / Warning / TODO: currently, the shiny.space IP is specified in the jitsi config - if swisscom gives me a new one i will need to update it manually. Will add monitoring for it and write a script sometime to automate.

Note about rootless setup:
if you look at the compose file, you might see that this container doesn't use "userns-mode: keep-id" or the x-podman: in_pod: false settings we had to use for other rootles containers to fix permissions. Why this is different from other containers, i do not know - fact is we can edit the configs. BUT some data is still owned by another user, so when it comes to backup or migration stuff we need to deal with it (or just delete everything, for now the plan is that this is "ethereal" content, it can disappear any time).

Jitsi compose.yml

```
version: '3.5'

services:
    # Frontend
    web:
        image: jitsi/web:${JITSI_IMAGE_VERSION:-stable-9753}
        restart: ${RESTART_POLICY:-unless-stopped}
        ports:
            - '${HTTP_PORT}:80'
            - '${HTTPS_PORT}:443'
        volumes:
            - ${CONFIG}/web:/config:Z
            - ${CONFIG}/web/crontabs:/var/spool/cron/crontabs:Z
            - ${CONFIG}/transcripts:/usr/share/jitsi-meet/transcripts:Z
        labels:
            service: "jitsi-web"
        environment:
            - AMPLITUDE_ID
            - ANALYTICS_SCRIPT_URLS
            - ANALYTICS_WHITELISTED_EVENTS
            - AUDIO_QUALITY_OPUS_BITRATE
            - AUTO_CAPTION_ON_RECORD
            - BRANDING_DATA_URL
            - BOSH_RELATIVE
            - CHROME_EXTENSION_BANNER_JSON
            - COLIBRI_WEBSOCKET_PORT
            - COLIBRI_WEBSOCKET_JVB_LOOKUP_NAME
            - COLIBRI_WEBSOCKET_REGEX
            - CONFCODE_URL
            - CORS_HEADER_ACCESS_CONTROL_ALLOW_ORIGIN
            - DEFAULT_LANGUAGE
            - DEPLOYMENTINFO_ENVIRONMENT
            - DEPLOYMENTINFO_ENVIRONMENT_TYPE
            - DEPLOYMENTINFO_REGION
```

```
      - DEPLOYMENTINFO_SHARD
      - DEPLOYMENTINFO_USERREGION
      - DESKTOP_SHARING_FRAMERATE_AUTO
      - DESKTOP_SHARING_FRAMERATE_MIN
      - DESKTOP_SHARING_FRAMERATE_MAX
      - DIALIN_NUMBERS_URL
      - DIALOUT_AUTH_URL
      - DIALOUT_CODES_URL
      - DISABLE_AUDIO_LEVELS
      - DISABLE_COLIBRI_WEBSOCKET_JVB_LOOKUP
      - DISABLE_DEEP_LINKING
      - DISABLE_GRANT_MODERATOR
      - DISABLE_HTTPS
      - DISABLE_KICKOUT
      - DISABLE_LOCAL_RECORDING
      - DISABLE_POLLS
      - DISABLE_PRIVATE_CHAT
      - DISABLE_PROFILE
      - DISABLE_REACTIONS
      - DISABLE_REMOTE_VIDEO_MENU
      - DISABLE_START_FOR_ALL
      - DROPBOX_APPKEY
      - DROPBOX_REDIRECT_URI
      - DYNAMIC_BRANDING_URL
      - ENABLE_AUDIO_PROCESSING
      - ENABLE_AUTH
      - ENABLE_AUTH_DOMAIN
      - ENABLE_BREAKOUT_ROOMS
      - ENABLE_CALENDAR
      - ENABLE_COLIBRI_WEBSOCKET
      - ENABLE_COLIBRI_WEBSOCKET_UNSAFE_REGEX
      - ENABLE_E2EPING
      - ENABLE_FILE_RECORDING_SHARING
      - ENABLE_GUESTS
      - ENABLE_HSTS
      - ENABLE_HTTP_REDIRECT
      - ENABLE_IPV6
      - ENABLE_LETSENCRYPT=${ENABLE_LETSENCRYPT}
      - ENABLE_NO_AUDIO_DETECTION
      - ENABLE_NOISY_MIC_DETECTION
```

```
        - ENABLE_OCTO
        - ENABLE_OPUS_RED
        - ENABLE_PREJOIN_PAGE
        - ENABLE_P2P
        - ENABLE_WELCOME_PAGE
        - ENABLE_CLOSE_PAGE
        - ENABLE_LIVESTREAMING
        - ENABLE_LIVESTREAMING_DATA_PRIVACY_LINK
        - ENABLE_LIVESTREAMING_HELP_LINK
        - ENABLE_LIVESTREAMING_TERMS_LINK
        - ENABLE_LIVESTREAMING_VALIDATOR_REGEXP_STRING
        - ENABLE_LOAD_TEST_CLIENT
        - ENABLE_LOCAL_RECORDING_NOTIFY_ALL_PARTICIPANT
        - ENABLE_LOCAL_RECORDING_SELF_START
        - ENABLE_RECORDING
        - ENABLE_REMB
        - ENABLE_REQUIRE_DISPLAY_NAME
        - ENABLE_SERVICE_RECORDING
        - ENABLE_SIMULCAST
        - ENABLE_STATS_ID
        - ENABLE_STEREO
        - ENABLE_SUBDOMAINS
        - ENABLE_TALK_WHILE_MUTED
        - ENABLE_TCC
        - ENABLE_TRANSCRIPTIONS
        - ENABLE_XMPP_WEBSOCKET
        - ENABLE_JAAS_COMPONENTS
        - ETHERPAD_PUBLIC_URL
        - ETHERPAD_URL_BASE
        - E2EPING_NUM_REQUESTS
        - E2EPING_MAX_CONFERENCE_SIZE
        - E2EPING_MAX_MESSAGE_PER_SECOND
        - GOOGLE_ANALYTICS_ID
        - GOOGLE_API_APP_CLIENT_ID
        - HIDE_PREMEETING_BUTTONS
        - HIDE_PREJOIN_DISPLAY_NAME
        - HIDE_PREJOIN_EXTRA_BUTTONS
        - INVITE_SERVICE_URL
        - JVB_PREFER_SCTP
        - LETSENCRYPT_DOMAIN
```

```
      - LETSENCRYPT_EMAIL
      - LETSENCRYPT_USE_STAGING
      - MATOMO_ENDPOINT
      - MATOMO_SITE_ID
      - MICROSOFT_API_APP_CLIENT_ID
      - NGINX_KEEPALIVE_TIMEOUT
      - NGINX_RESOLVER
      - NGINX_WORKER_PROCESSES
      - NGINX_WORKER_CONNECTIONS
      - PEOPLE_SEARCH_URL
      - PREFERRED_LANGUAGE
      - PUBLIC_URL=${PUBLIC_URL}
      - P2P_PREFERRED_CODEC
      - RESOLUTION
      - RESOLUTION_MIN
      - RESOLUTION_WIDTH
      - RESOLUTION_WIDTH_MIN
      - START_AUDIO_MUTED
      - START_AUDIO_ONLY
      - START_SILENT
      - START_WITH_AUDIO_MUTED
      - START_VIDEO_MUTED
      - START_WITH_VIDEO_MUTED
      - TESTING_AV1_SUPPORT
      - TOKEN_AUTH_URL
      - TOOLBAR_BUTTONS
      - TRANSLATION_LANGUAGES
      - TRANSLATION_LANGUAGES_HEAD
      - TZ=${TZ}
      - USE_APP_LANGUAGE
      - VIDEOQUALITY_BITRATE_H264_LOW
      - VIDEOQUALITY_BITRATE_H264_STANDARD
      - VIDEOQUALITY_BITRATE_H264_HIGH
      - VIDEOQUALITY_BITRATE_H264_FULL
      - VIDEOQUALITY_BITRATE_H264_ULTRA
      - VIDEOQUALITY_BITRATE_H264_SS_HIGH
      - VIDEOQUALITY_BITRATE_VP8_LOW
      - VIDEOQUALITY_BITRATE_VP8_STANDARD
      - VIDEOQUALITY_BITRATE_VP8_HIGH
      - VIDEOQUALITY_BITRATE_VP8_FULL
```

```
        - VIDEOQUALITY_BITRATE_VP8_ULTRA
        - VIDEOQUALITY_BITRATE_VP8_SS_HIGH
        - VIDEOQUALITY_BITRATE_VP9_LOW
        - VIDEOQUALITY_BITRATE_VP9_STANDARD
        - VIDEOQUALITY_BITRATE_VP9_HIGH
        - VIDEOQUALITY_BITRATE_VP9_FULL
        - VIDEOQUALITY_BITRATE_VP9_ULTRA
        - VIDEOQUALITY_BITRATE_VP9_SS_HIGH
        - VIDEOQUALITY_BITRATE_AV1_LOW
        - VIDEOQUALITY_BITRATE_AV1_STANDARD
        - VIDEOQUALITY_BITRATE_AV1_HIGH
        - VIDEOQUALITY_BITRATE_AV1_FULL
        - VIDEOQUALITY_BITRATE_AV1_ULTRA
        - VIDEOQUALITY_BITRATE_AV1_SS_HIGH
        - VIDEOQUALITY_PREFERRED_CODEC
        - XMPP_AUTH_DOMAIN
        - XMPP_BOSH_URL_BASE
        - XMPP_DOMAIN
        - XMPP_GUEST_DOMAIN
        - XMPP_MUC_DOMAIN
        - XMPP_RECORDER_DOMAIN
        - XMPP_PORT
        - WHITEBOARD_COLLAB_SERVER_PUBLIC_URL
        - WHITEBOARD_COLLAB_SERVER_URL_BASE
    networks:
        meet.jitsi:
    depends_on:
        - jvb

# XMPP server
prosody:
    image: jitsi/prosody:${JITSI_IMAGE_VERSION:-stable-9753}
    restart: ${RESTART_POLICY:-unless-stopped}
    expose:
        - '${XMPP_PORT:-5222}'
        - '${PROSODY_S2S_PORT:-5269}'
        - '5347'
        - '${PROSODY_HTTP_PORT:-5280}'
    labels:
        service: "jitsi-prosody"
```

```
    volumes:
        - ${CONFIG}/prosody/config:/config:Z
        - ${CONFIG}/prosody/prosody-plugins-custom:/prosody-plugins-custom:Z
    environment:
        - AUTH_TYPE
        - DISABLE_POLLS
        - ENABLE_AUTH
        - ENABLE_AV_MODERATION
        - ENABLE_BREAKOUT_ROOMS
        - ENABLE_END_CONFERENCE
        - ENABLE_GUESTS
        - ENABLE_IPV6
        - ENABLE_LOBBY
        - ENABLE_RECORDING
        - ENABLE_S2S
        - ENABLE_TRANSCRIPTIONS
        - ENABLE_VISITORS
        - ENABLE_XMPP_WEBSOCKET
        - ENABLE_JAAS_COMPONENTS
        - GC_TYPE
        - GC_INC_TH
        - GC_INC_SPEED
        - GC_INC_STEP_SIZE
        - GC_GEN_MIN_TH
        - GC_GEN_MAX_TH
        - GLOBAL_CONFIG
        - GLOBAL_MODULES
        - JIBRI_RECORDER_USER
        - JIBRI_RECORDER_PASSWORD=${JIBRI_RECORDER_PASSWORD}
        - JIBRI_SIP_BREWERY_MUC
        - JIBRI_XMPP_USER
        - JIBRI_XMPP_PASSWORD=${JIBRI_XMPP_PASSWORD}
        - JICOFO_AUTH_PASSWORD=${JICOFO_AUTH_PASSWORD}
        - JICOFO_COMPONENT_SECRET
        - JIGASI_TRANSCRIBER_PASSWORD=${JIGASI_TRANSCRIBER_PASSWORD}
        - JIGASI_TRANSCRIBER_USER
        - JIGASI_XMPP_USER
        - JIGASI_XMPP_PASSWORD=${JIGASI_XMPP_PASSWORD}
        - JVB_AUTH_USER
        - JVB_AUTH_PASSWORD=${JVB_AUTH_PASSWORD}
```

```
      - JWT_APP_ID
      - JWT_APP_SECRET
      - JWT_ACCEPTED_ISSUERS
      - JWT_ACCEPTED_AUDIENCES
      - JWT_ASAP_KEYSERVER
      - JWT_ALLOW_EMPTY
      - JWT_AUTH_TYPE
      - JWT_ENABLE_DOMAIN_VERIFICATION
      - JWT_SIGN_TYPE
      - JWT_TOKEN_AUTH_MODULE
      - MATRIX_UVS_URL
      - MATRIX_UVS_ISSUER
      - MATRIX_UVS_AUTH_TOKEN
      - MATRIX_UVS_SYNC_POWER_LEVELS
      - MATRIX_LOBBY_BYPASS
      - LOG_LEVEL
      - LDAP_AUTH_METHOD
      - LDAP_BASE
      - LDAP_BINDDN
      - LDAP_BINDPW
      - LDAP_FILTER
      - LDAP_VERSION
      - LDAP_TLS_CIPHERS
      - LDAP_TLS_CHECK_PEER
      - LDAP_TLS_CACERT_FILE
      - LDAP_TLS_CACERT_DIR
      - LDAP_START_TLS
      - LDAP_URL
      - LDAP_USE_TLS
      - MAX_PARTICIPANTS
      - PROSODY_ADMINS
      - PROSODY_AUTH_TYPE
      - PROSODY_C2S_LIMIT
      - PROSODY_C2S_REQUIRE_ENCRYPTION
      - PROSODY_RESERVATION_ENABLED
      - PROSODY_RESERVATION_REST_BASE_URL
      - PROSODY_ENABLE_RATE_LIMITS
      - PROSODY_ENABLE_RECORDING_METADATA
      - PROSODY_ENABLE_STANZA_COUNTS
      - PROSODY_ENABLE_S2S
```

```
      - PROSODY_ENABLE_METRICS
      - PROSODY_GUEST_AUTH_TYPE
      - PROSODY_HTTP_PORT
      - PROSODY_LOG_CONFIG
      - PROSODY_METRICS_ALLOWED_CIDR
      - PROSODY_MODE
      - PROSODY_RATE_LIMIT_LOGIN_RATE
      - PROSODY_RATE_LIMIT_SESSION_RATE
      - PROSODY_RATE_LIMIT_TIMEOUT
      - PROSODY_RATE_LIMIT_ALLOW_RANGES
      - PROSODY_RATE_LIMIT_CACHE_SIZE
      - PROSODY_S2S_LIMIT
      - PROSODY_S2S_PORT
      - PROSODY_TRUSTED_PROXIES
      - PROSODY_VISITOR_INDEX
      - PROSODY_VISITORS_MUC_PREFIX
      - PUBLIC_URL=${PUBLIC_URL}
      - STUN_HOST
      - STUN_PORT
      - TURN_CREDENTIALS
      - TURN_HOST
      - TURNS_HOST
      - TURN_PORT
      - TURNS_PORT
      - TURN_TRANSPORT
      - TZ=${TZ}
      - VISITORS_MAX_VISITORS_PER_NODE
      - VISITORS_XMPP_DOMAIN
      - VISITORS_XMPP_SERVER
      - VISITORS_XMPP_PORT
      - XMPP_BREAKOUT_MUC_MODULES
      - XMPP_CONFIGURATION
      - XMPP_DOMAIN
      - XMPP_AUTH_DOMAIN
      - XMPP_GUEST_DOMAIN
      - XMPP_MUC_DOMAIN
      - XMPP_INTERNAL_MUC_DOMAIN
      - XMPP_LOBBY_MUC_MODULES
      - XMPP_MODULES
      - XMPP_MUC_MODULES
```

```
        - XMPP_MUC_CONFIGURATION
        - XMPP_INTERNAL_MUC_MODULES
        - XMPP_RECORDER_DOMAIN
        - XMPP_PORT
        - XMPP_SERVER_S2S_PORT
        - XMPP_SPEAKERSTATS_MODULES
    networks:
        meet.jitsi:
            aliases:
                - ${XMPP_SERVER:-xmpp.meet.jitsi}

# Focus component
jicofo:
    image: jitsi/jicofo:${JITSI_IMAGE_VERSION:-stable-9753}
    restart: ${RESTART_POLICY:-unless-stopped}
    ports:
        - '127.0.0.1:${JICOFO_REST_PORT:-8888}:8888'
    volumes:
        - ${CONFIG}/jicofo:/config:Z
    labels:
        service: "jitsi-jicofo"
    environment:
        - AUTH_TYPE
        - BRIDGE_AVG_PARTICIPANT_STRESS
        - BRIDGE_STRESS_THRESHOLD
        - ENABLE_AUTH
        - ENABLE_AUTO_OWNER
        - ENABLE_CODEC_VP8
        - ENABLE_CODEC_VP9
        - ENABLE_CODEC_AV1
        - ENABLE_CODEC_H264
        - ENABLE_CODEC_OPUS_RED
        - ENABLE_JVB_XMPP_SERVER
        - ENABLE_OCTO
        - ENABLE_OCTO_SCTP
        - ENABLE_RECORDING
        - ENABLE_SCTP
        - ENABLE_TRANSCRIPTIONS
        - ENABLE_VISITORS
        - ENABLE_AUTO_LOGIN
```

```
      - JICOFO_AUTH_LIFETIME
      - JICOFO_AUTH_PASSWORD=${JICOFO_AUTH_PASSWORD}
      - JICOFO_AUTH_TYPE
      - JICOFO_BRIDGE_REGION_GROUPS
      - JICOFO_ENABLE_AUTH
      - JICOFO_ENABLE_BRIDGE_HEALTH_CHECKS
      - JICOFO_CONF_INITIAL_PARTICIPANT_WAIT_TIMEOUT
      - JICOFO_CONF_SINGLE_PARTICIPANT_TIMEOUT
      - JICOFO_CONF_SOURCE_SIGNALING_DELAYS
      - JICOFO_CONF_MAX_AUDIO_SENDERS
      - JICOFO_CONF_MAX_VIDEO_SENDERS
      - JICOFO_CONF_STRIP_SIMULCAST
      - JICOFO_CONF_SSRC_REWRITING
      - JICOFO_ENABLE_HEALTH_CHECKS
      - JICOFO_ENABLE_REST
      - JICOFO_HEALTH_CHECKS_USE_PRESENCE
      - JICOFO_MAX_MEMORY
      - JICOFO_MULTI_STREAM_BACKWARD_COMPAT
      - JICOFO_OCTO_REGION
      - JICOFO_TRUSTED_DOMAINS
      - JIBRI_BREWERY_MUC
      - JIBRI_REQUEST_RETRIES
      - JIBRI_PENDING_TIMEOUT
      - JIGASI_BREWERY_MUC
      - JIGASI_SIP_URI
      - JIGASI_TRUSTED_DOMAINS
      - JVB_BREWERY_MUC
      - JVB_XMPP_AUTH_DOMAIN
      - JVB_XMPP_INTERNAL_MUC_DOMAIN
      - JVB_XMPP_PORT
      - JVB_XMPP_SERVER
      - MAX_BRIDGE_PARTICIPANTS
      - OCTO_BRIDGE_SELECTION_STRATEGY
      - PROSODY_VISITORS_MUC_PREFIX
      - SENTRY_DSN="${JICOFO_SENTRY_DSN:-0}"
      - SENTRY_ENVIRONMENT
      - SENTRY_RELEASE
      - TZ=${TZ}
      - VISITORS_MAX_PARTICIPANTS
      - VISITORS_MAX_VISITORS_PER_NODE
```

```
        - VISITORS_XMPP_AUTH_DOMAIN
        - VISITORS_XMPP_SERVER
        - VISITORS_XMPP_DOMAIN
        - XMPP_DOMAIN
        - XMPP_AUTH_DOMAIN
        - XMPP_INTERNAL_MUC_DOMAIN
        - XMPP_MUC_DOMAIN
        - XMPP_RECORDER_DOMAIN
        - XMPP_SERVER
        - XMPP_PORT
        - MAX_SSRCS_PER_USER
        - MAX_SSRC_GROUPS_PER_USER
    depends_on:
        - prosody
    networks:
        meet.jitsi:

# Video bridge
jvb:
    image: jitsi/jvb:${JITSI_IMAGE_VERSION:-stable-9753}
    restart: ${RESTART_POLICY:-unless-stopped}
    ports:
        - '${JVB_PORT:-10000}:${JVB_PORT:-10000}/udp'
        - '127.0.0.1:${JVB_COLIBRI_PORT:-8080}:8080'
    volumes:
        - ${CONFIG}/jvb:/config:Z
    labels:
        service: "jitsi-jvb"
    environment:
        - AUTOSCALER_SIDECAR_KEY_FILE
        - AUTOSCALER_SIDECAR_KEY_ID
        - AUTOSCALER_SIDECAR_GROUP_NAME
        - AUTOSCALER_SIDECAR_HOST_ID
        - AUTOSCALER_SIDECAR_INSTANCE_ID
        - AUTOSCALER_SIDECAR_PORT
        - AUTOSCALER_SIDECAR_REGION
        - AUTOSCALER_SIDECAR_SHUTDOWN_POLLING_INTERVAL
        - AUTOSCALER_SIDECAR_STATS_POLLING_INTERVAL
        - DOCKER_HOST_ADDRESS
        - ENABLE_COLIBRI_WEBSOCKET
```

```
      - ENABLE_JVB_XMPP_SERVER
      - ENABLE_OCTO
      - ENABLE_SCTP
      - JVB_ADVERTISE_IPS=${JVB_ADVERTISE_IPS}
      - JVB_ADVERTISE_PRIVATE_CANDIDATES
      - JVB_AUTH_USER
      - JVB_AUTH_PASSWORD=${JVB_AUTH_PASSWORD}
      - JVB_BREWERY_MUC
      - JVB_CC_TRUST_BWE
      - JVB_DISABLE_STUN
      - JVB_DISABLE_XMPP
      - JVB_INSTANCE_ID
      - JVB_PORT
      - JVB_MUC_NICKNAME
      - JVB_STUN_SERVERS
      - JVB_LOG_FILE
      - JVB_OCTO_BIND_ADDRESS
      - JVB_OCTO_REGION
      - JVB_OCTO_RELAY_ID
      - JVB_REQUIRE_VALID_ADDRESS
      - JVB_USE_USRSCTP
      - JVB_WS_DOMAIN
      - JVB_WS_SERVER_ID
      - JVB_WS_TLS
      - JVB_XMPP_AUTH_DOMAIN
      - JVB_XMPP_INTERNAL_MUC_DOMAIN
      - JVB_XMPP_PORT
      - JVB_XMPP_SERVER
      - PUBLIC_URL=${PUBLIC_URL}
      - SENTRY_DSN="${JVB_SENTRY_DSN:-0}"
      - SENTRY_ENVIRONMENT
      - SENTRY_RELEASE
      - COLIBRI_REST_ENABLED
      - SHUTDOWN_REST_ENABLED
      - TZ=${TZ}
      - VIDEOBRIDGE_MAX_MEMORY
      - XMPP_AUTH_DOMAIN
      - XMPP_INTERNAL_MUC_DOMAIN
      - XMPP_SERVER
      - XMPP_PORT
```

```
        depends_on:
            - prosody
        networks:
            meet.jitsi:

# Custom network so all services can communicate using a FQDN
networks:
    meet.jitsi:
```

nginx reverse proxy configuration

/etc/nginx/sites-available/meet.shiny.space

(why debian?? we don't need sites-available for nginx... switch to alpine sometime)

```
server {
	listen 443 ssl;

	server_name meet.shiny.space;
	client_max_body_size 500M;
	location /.well-known {
		root /var/www/html;
		try_files $uri $uri/ = 404;
	}
	location /xmpp-websocket {
		proxy_pass http://10.0.0.30:8000;
    		proxy_http_version 1.1;
    		proxy_set_header Upgrade $http_upgrade;
		proxy_set_header Connection "upgrade";
	}
	location /colibri-ws {
		proxy_pass http://10.0.0.30:8000;
		proxy_http_version 1.1;
		proxy_set_header Upgrade $http_upgrade;
		proxy_set_header Connection "upgrade";
	}
	location /http-bind {
		proxy_pass http://10.0.0.30:8000;
		proxy_http_version 1.1;
		proxy_set_header Upgrade $http_upgrade;
		proxy_set_header Connection "upgrade";
	}
	location / {
```

```
□□proxy_pass http://10.0.0.30:8000;
□}
}
```

Config - ENV and interface customization

Most important is the ENV file - here's the currently set variables. if jitsi sees some usage, we'll add the optional services too.

```
# config location changed to keep everything in one directory
CONFIG=~/jitsi/.jitsi-meet-cfg

# Exposed HTTP port (will redirect to HTTPS port)
HTTP_PORT=8000

# Exposed HTTPS port
HTTPS_PORT=8443

# custom: jvb colibri port, defaults to 8080 otherwise
JVB_COLIBRI_PORT=8585

# System time zone
TZ=CEST

# Public URL for the web service (required)
# Keep in mind that if you use a non-standard HTTPS port, it has to appear in the public URL
PUBLIC_URL=https://meet.shiny.space

# Media IP addresses to advertise by the JVB
# This setting deprecates DOCKER_HOST_ADDRESS, and supports a comma separated list of IPs
# See the "Running behind NAT or on a LAN environment" section in the Handbook:
# https://jitsi.github.io/handbook/docs/devops-guide/devops-guide-docker#running-behind-nat-or-on-a-lan-environment
# NOTE BY LUCAS: CHANGE THIS IF PUBLIC IP CHANGES
JVB_ADVERTISE_IPS=83.76.33.165

# Enable Let's Encrypt certificate generation
# NOTE BY LUCAS: disable because currently nginx reverse proxy handles all HTTPS certs
```

```
ENABLE_LETSENCRYPT=0

#
# Security
#
# Set these to strong passwords to avoid intruders from impersonating a service account
# The service(s) won't start unless these are specified
# Running ./gen-passwords.sh will update .env with strong passwords
# You may skip the Jigasi and Jibri passwords if you are not using those
# DO NOT reuse passwords
#

# Here would be the passwords, obviously not in bookstack.
JICOFO_AUTH_PASSWORD=
JVB_AUTH_PASSWORD=
.
.
.
```

then a few customizations for the web interface. by default Jitsi has a watermark with a big link to jitsi.org, visible at all times during the call and accidentally clickable too. we disable it, and we also disable the "install the app" prompt on mobile - although the app is fully open source and available on f-droid! it's really not bad.

To make the configuration, we can put a file called custom-interface_config.js in the proper location, and when restarting the container it automatically inserts the new config into the default interface_config.js. Note: do not edit interface_config.js directly, it is overwritten each time the container starts!

~/jitsi/.jitsi-meet-cfg/web/custom-interface_config.js:

```
interfaceConfig.JITSI_WATERMARK_LINK = '';
interfaceConfig.SHOW_JITSI_WATERMARK = false;
interfaceConfig.MOBILE_APP_PROMO = false;
```

Note: the file says it is deprecated and all options will be moved to config.js in the future. keep this in mind when updating, at some point the settings will have to be moved. likely just to custom-config.js and config.MYVALUE instead of interfaceConfig.MYVALUE.

TODO: maybe increase max video quality? set defaults for joining rooms - not let anyone join at will? setup accounts?

STUN / TURN server

Some networks have complicated NAT in front of them which can make it a little difficult to open direct connections using Websockets (like jitsi does) or similar methods between two devices. Also, some firewalls can be very strict about what connections they allow, which can lead to connection issues in Jitsi.

For these cases, we have two solutions:
the STUN server is a public server with no NAT before it at all, it's just a publicly reachable server with a public, fixed (statically assigned) IP address. so when two clients have issues getting through NATs and firewalls because of the nature of their networks, they can both connect to the STUN server, which makes the initial communication between the two easier and can allow them to reach each other directly through some very smart methods. Here's a quick summary from a youtube commend on STUN:

> The main hole punching technique was not explained here. It works as follows:
> - A and B gather the public IP and port of each other through a third party server.
> - A and B start sending packets to each other. Let's say A went first.
> - Packet from A to B is blocked by B's NAT because it was not a response.
> - Packet from B to A was sent to A because its NAT identified it as a response to the above packet.
> - Packet from A to B is also identified as a response. Hole punching is achieved.

i have read a nicer explanation somewhere but can't find it any more.

The second solution is TURN - pretty much just routing the entire traffic through a TURN server. this costs bandwidth, so there's usually no public, open TURN servers - we set up our own just in case someone is behind really strict NAT or firewalls.

To run a STUN / TURN server, we install coturn on our vps running debian, and put the following configs in /etc/stunserver.conf

```
# summary, the actual file contains way more, but the defaults are usually good


# TURN listener port for UDP and TCP (Default: 3478).
# Note: actually, TLS & DTLS sessions can connect to the
# "plain" TCP & UDP port(s), too - if allowed by configuration.
#
```

```
listening-port=3478

# TURN listener port for TLS (Default: 5349).
# Note: actually, "plain" TCP & UDP sessions can connect to the TLS & DTLS
# port(s), too - if allowed by configuration. The TURN server
# "automatically" recognizes the type of traffic. Actually, two listening
# endpoints (the "plain" one and the "tls" one) are equivalent in terms of
# functionality; but Coturn keeps both endpoints to satisfy the RFC 5766 specs.
# For secure TCP connections, Coturn currently supports
# TLS version 1.0, 1.1 and 1.2.
# For secure UDP connections, Coturn supports DTLS version 1.
#
tls-listening-port=5349



# Listener IP address of relay server. Multiple listeners can be specified.
# If no IP(s) specified in the config file or in the command line options,
# then all IPv4 and IPv6 system IPs will be used for listening.
#
#listening-ip=172.17.19.101
#listening-ip=10.207.21.238
#listening-ip=2607:f0d0:1002:51::4
listening-ip=65.108.90.108
listening-ip=2a01:4f9:c010:d401::1
# note by lucas: these are the current public ipv4 / ipv6 addresses of the Hetzner vps.

# Uncomment to use long-term credential mechanism.
# By default no credentials mechanism is used (any user allowed).
#
lt-cred-mech

# 'Static' user accounts for the long term credentials mechanism, only.
# This option cannot be used with TURN REST API.
# 'Static' user accounts are NOT dynamically checked by the turnserver process,
# so they can NOT be changed while the turnserver is running.
#
#user=username1:key1
#user=username2:key2
# OR:
#user=username1:password1
```

```
#user=username2:password2
#
# Keys must be generated by turnadmin utility. The key value depends
# on user name, realm, and password:
#
# Example:
# $ turnadmin -k -u ninefingers -r north.gov -p youhavetoberealistic
# Output: 0xbc807ee29df3c9ffa736523fb2c4e8ee
# ('0x' in the beginning of the key is what differentiates the key from
# password. If it has 0x then it is a key, otherwise it is a password).
#
# The corresponding user account entry in the config file will be:
#
#user=ninefingers:0xbc807ee29df3c9ffa736523fb2c4e8ee
# Or, equivalently, with open clear password (less secure):
#user=ninefingers:youhavetoberealistic
#
user=shinyjitsi:<password>


# The default realm to be used for the users when no explicit
# origin/realm relationship is found in the database, or if the TURN
# server is not using any database (just the commands-line settings
# and the userdb file). Must be used with long-term credentials
# mechanism or with TURN REST API.
#
# Note: If the default realm is not specified, then realm falls back to the host domain name.
#       If the domain name string is empty, or set to '(None)', then it is initialized as an
empty string.
#
realm=vps.shiny.space


# Option to hide software version. Enhance security when used in production.
# Revealing the specific software version of the agent through the
# SOFTWARE attribute might allow them to become more vulnerable to
# attacks against software that is known to contain security holes.
# Implementers SHOULD make usage of the SOFTWARE attribute a
# configurable option (https://tools.ietf.org/html/rfc5389#section-16.1.2)
#
```

```
no-software-attribute
```

then start the service and allow the specified ports in the firewall (here ufw)