

2024/09/06 Gfarmシンポジウム2024

Gfarmのクラウドデプロイと Lustre HSM連携

株式会社 創夢

石橋拓也

(Gfarm経験 2002年～)

本日の内容 (2本立て)

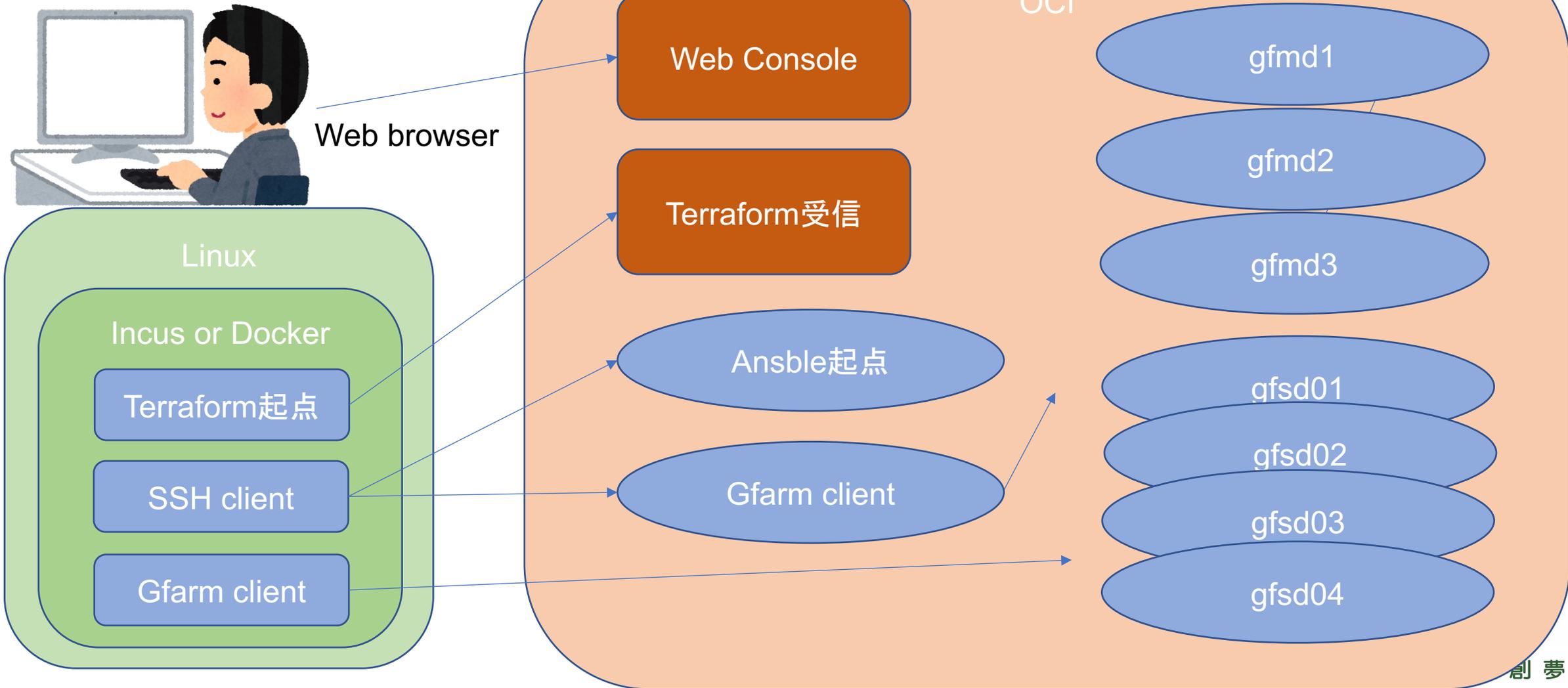
- Gfarmのクラウドデプロイ
 - Oracle Cloud InfrastructureにGfarmをデプロイする仕組み紹介
- GfarmのLustre HSM連携
 - Lustre HSM (Hierarchical Storage Management) 概要
 - Lustre HSMとしてGfarmにアーカイブする仕組み紹介
 - お試し環境自動構築手順

Gfarmのクラウドデプロイ

Gfarmのクラウドデプロイ概要

- デプロイスクリプト一式
 - <https://github.com/oss-tsukuba/incus-auto/tree/main/example/gfarm-terraform-oci>
 - (今後のメンテ方針:未定)
- Oracle Cloud Infrastructure (OCI) にGfarm環境をデプロイする
- TerraformとAnsible使用
- IncusまたはDocker使用
 - 手元の環境を汚さない
- Oracle Cloudにアカウントと課金が必要
 - コスト試算 <https://www.oracle.com/jp/cloud/costestimator.html>
 - 無料枠があるが、無料枠内では試せないかも(?)

Gfarmのクラウドデプロイ 構成図



Gfarmのクラウドデプロイ構成 (1)

- OCIのWebコンソール操作のためのWebブラウザ
 - 各設定を事前に行う、各パラメータをコピーで取得
- デプロイ操作起点とするLinuxマシン (手元のPCでも可)
 - IncusまたはDockerコンテナ内にOCI制御起点を構築
 - メモリ使用量は少ない
 - SSHでOCIのインスタンスに接続
 - Gfarmクライアントとしても利用可能
- Terraformの定義・設定ファイル
 - OCI向けの定義を記載 (TerraformのOCI用プロバイダ説明書を参考)
 - <https://registry.terraform.io/providers/oracle/oci/latest/docs>
 - 他のクラウド向け定義一式を作成して、スクリプトを微調整すれば、他でもGfarm環境を構築できる想定 (未確認)

Gfarmのクラウドデプロイ構成 (2)

- OCIに作成されるインスタンス
 - 管理ホスト (Ansibleの起点)
 - Gfarmクライアント x 1~任意数 (デフォルト1)
 - Gfarm gfmd x 1~任意数 (デフォルト3)
 - Gfarm gfsd x 1~任意数 (デフォルト4)
- OS: Oracle Linux 9
- シェイプ: VM.Standard.A1.Flex (Arm系プロセッサ)
 - OCPU数: 任意
 - メモリサイズ: 任意
 - ディスクサイズ: 任意

(補足) Terraformとは

- クラウドのインスタンスやネットワークなどの構成をコードとして記載し、作成・変更・削除ができる
- プロバイダーごとのプラグインを使い分けることで、さまざまなクラウドに対応する
 - 各プロバイダーごとに定義方法が異なる
 - プロバイダ間の違いを吸収・抽象化する仕組みは無い
- 感想
 - 繰り返し処理の記述に慣れず苦戦した

(補足) Ansibleとは

- AnsibleとSSHクライアントを使って、リモートマシンのシステムやアプリケーション構成を設定・状態維持できる
- 各リモートマシン側にAnsible特化のアプリは不要
- さまざまなモジュールが用意されているので、よく使う操作は、簡潔に定義できる

- 感想
 - Gfarmの設定ファイルのためにリモートサーバ間のファイルコピーをしたかったが、できないので、Ansible起点経由でコピーした

(補足)

Incusとは (1)

- <https://linuxcontainers.org/incus/>
- システムコンテナ(LXCベース)やVM(QEMUベース)と、そのためのネットワークを作成・管理できる
- 操作コマンドがシンプル
- Incusコンテナは、VMで動かしたLinuxサーバーのように利用可能
 - コンテナの場合、ホストOSのカーネルを使用
 - コンテナの場合、VMよりも、1インスタンスあたりの使用メモリ、使用ストレージが少ない
 - Dockerのようなアプリコンテナとは別の仕組み、用途 (使い分け)
 - VMのような環境をDockerコンテナのような軽さで利用可能
 - Dockerイメージ自体も最近Incusで実行できるようになった
- 公式で配布されているコンテナ・VMイメージ
 - <https://images.linuxcontainers.org/>

(補足) Incusとは (2)

- 操作例
 - `$ incus launch images:rockylinux/9 rocky9 [--vm]`
(イメージダウンロード時間[キャッシュされる]+起動時間[短時間])
 - `$ incus shell rocky9`
 - `[root@rocky9 ~]# logout`
 - `$ incus stop rocky9`
 - `$ incus delete rocky9`
- 2023年、LXDからフォークした
 - LXDの元開発者がIncus開発に関わっている
 - 現在、コマンド体系はほぼ同じ
 - 今後互換性が無くなる可能性はある
 - 現在、LXDからIncusのイメージサーバを利用不可

(補足) incus-autoを実装した

- <https://github.com/oss-tsukuba/incus-auto>
- Incusのインスタンス・ネットワーク構成をYAMLファイルで定義して環境構築する仕組み
 - 構成定義に従ってincusコマンドを呼び出し
- Docker Composeのような使用感を目指した
 - incus-autoを使わなくても、通常は、シェルスクリプトからincusコマンド呼び出しや、Terraform (またはOpenTofu) を使ってIncusを制御すれば良い
 - incus-autoから利用できないIncusの機能はまだ多数ある
 - (作り込み未定)
- Incusだけでなく、LXDでも利用可能

Gfarmのクラウドデプロイ 手順概要

- Step 1: 設定など
 - 構築スクリプト取得、Gfarm関連ソースコードを取得、incus-auto準備
 - OCI用APIキーとSSH鍵ペアを手元に生成
 - OCIのWebコンソールでAPIキー, VCN, Ingressルールなどを設定
 - 各OCID値を手元の設定ファイルにコピー&ペースト
- Step 2: Terraform環境準備
 - IncusまたはDockerテナ内にTerraform実行環境を構築
- Step 3: TerraformでOCIインスタンス作成
 - Terraformを使ってOCIにインスタンス生成
 - 必要なファイル一式をOCIのインスタンスに転送
- Step 4: AnsibleでGfarm環境構築
 - OCIのインスタンスにSSHでログイン
 - AnsibleでGfarm環境を構築

Gfarmのクラウドデプロイ

Step 1: 設定など (1)

- gfarm-terraform-ociデプロイスクリプト一式取得
 - \$ git clone <https://github.com/oss-tsukuba/incus-auto.git>
 - incus-auto/example/gfarm-terraform-oci を使う
- IncusまたはDockerが動作するLinux環境を用意
 - Incusのインストール手順 (UbuntuやFedoraなど)
 - <https://linuxcontainers.org/incus/docs/main/installing/#installing>
 - RHEL系ではまだ動作しない
- Incus, incus-auto、またはDocker関連インストール
 - 詳細 incus-auto/README.md 参照
- GNU makeインストール
 - \$ sudo apt-get install make

Gfarmのクラウドデプロイ

Step 1: 設定など (2)

- `$ cd incus-auto/example/gfarm-terraform-oci`
- `$ make git-clone # Gfarm関連ソースコード取得`
- `$ make ssh-keygen`
- `$ make ociapi-keygen`
- `$ cat ./terraform/ociapi_public.pem # APIキーを表示`
- OCI Webコンソールで設定
 - APIキーの追加 (コピー&ペースト)
 - VCN・サブネットの作成 (必要なら)
 - セキュリティリスト イングレスルール (Firewall) の設定

APIキーの追加画面

ORACLE Cloud Cloud Classic > リソース、サービス、ドキュメントおよびマーケット: Japan East (Tokyo) > <> 🔔 ? 🌐 👤

ミドル・ネー
姓: Ishibashi
接尾辞: -

> ユーザー
> 勤務先情
> その他の

リソース

自分のグループ
統合アプリケーション
APIキー
認証トークン
顧客秘密キー

APIキーの追加

APIキーの追加

APIキーの追加

APIキーの追加

APIキーの追加

APIキーの追加 [ヘルプ](#)

ノート: APIキーは、APIリクエストの署名に使用されるPEM形式のRSAキー・ペアです。ここでキー・ペアを生成して秘密キーをダウンロードできます。すでにキー・ペアを保持している場合は、かわりに公開キー・ファイルをアップロードするか、貼り付けることを選択できます。 [さらに学ぶ](#)

APIキー・ペアの生成 公開キー・ファイルの選択 公開キーの貼付け

公開キー

インGRESルール

ソースアドレス範囲	IPプロトコル	宛先ポート範囲	用途
10.0.0.0/8 (例) (OCI内)	TCP	All	gfsd, sshd (インスタンス同士許可)
10.0.0.0/8 (例) (OCI内)	UDP	600	gfsd (インスタンス同士許可)
(Terraform起点, ssh接続元)	TCP	22	sshd
(Gfarm接続元)	TCP	601	gfmd
(Gfarm接続元)	TCP	600	gfsd
(Gfarm接続元)	UDP	600	gfsd

接続元IPアドレスと、OCI内サブネットのプライベートアドレス範囲を許可

Gfarmのクラウドデプロイ

Step 1: 設定など (3-1)

- ./terraform/terraform.tfvars ファイルを作成 (1)
 - 必須項目: Webコンソールより、各OCIDなどをコピー&ペースト
 - 取得先画面URLは ./terraform/variables.tf 参照

```
tenancy_ocid = "ocid1.tenancy.oc1..aaaaaaaa....."  
user_ocid = "ocid1.user.oc1..aaaaaaaa....."  
compartment_id = "ocid1.tenancy.oc1..aaaaaaaa....." # tenancy_ocidと同じ  
vcn_id = "ocid1.vcn.oc1.ap-tokyo-1.amaaaaaa....."  
subnet_id = "ocid1.subnet.oc1.ap-tokyo-1.aaaaaaaa....."  
view_id = "ocid1.dnsview.oc1.ap-tokyo-1.aaaaaaaa....." # プライベート・ビュー  
region = "ap-tokyo-1"
```

```
private_key_path = "ociapi_private.pem"  
fingerprint = "(APIキーのフィンガープリント)"  
ssh_authorized_keys = "id_ecdsa.pub"
```

Gfarmのクラウドデプロイ

Step 1: 設定など (3-2)

- ./terraform/terraform.tfvars ファイルを作成 (2)
 - 調整項目: デフォルト値は ./terraform/variables.tf 参照

```
domain = "(ドメイン名)"
gfmd_num = (gfmd台数)
gfsd_num = (gfsd台数)
gfclient_num = (Gfarm client台数)
gfmd_ocpus = (CPUコア数)
gfsd_ocpus = (CPUコア数)
gflient_ocpus = (CPUコア数)
gfmd_memory_in_gbs = (gfmdメモリサイズ)
gfsd_memory_in_gbs = (gfsdメモリサイズ)
gfclient_memory_in_gbs = (Gfarm clientメモリサイズ)
gfmd_volume_vpus_per_gb = (gfmdディスクサイズ)
gfsd_volume_vpus_per_gb = (gfsdディスクサイズ)
gfclient_volume_vpus_per_gb = (Gfarm clientディスクサイズ)
(その他)
```

Gfarmのクラウドデプロイ

Step 2: Terraform環境準備

- Incusを使う場合 (incus-auto使用)
 - Incusをインストール、セットアップしておく
 - \$ make incus-init
 - \$ make incus-create
 - \$ make shell
- Dockerを使う場合 (Docker Compose使用)
 - Dockerをインストール、セットアップしておく
 - 以降で実行するユーザ権限で docker コマンドを実行可能に設定しておく
 - \$ make docker-build
 - \$ make docker-up
 - \$ make docker-shell
- 以降、コンテナのシェルを利用 (コンテナ名: tf)

Gfarmのクラウドデプロイ

Step 3: TerraformでOCIインスタンス作成

- (tfコンテナにて)
- \$ cd ~/terraform
- \$ terraform init
- \$ terraform plan
- \$ terraform apply
- \$ make update-ssh_known_hosts
- \$ make update-ssh_config
- \$ make update-ansible-inventory
- \$ make send-files # 設定、ソースコード一式をOCI側にコピー

Gfarmのクラウドデプロイ

Step 4: AnsibleでGfarm環境構築

- (tfテナ) \$./ocissh gfmanage
- (OCI側 gfmanageホストにて)
- \$ cd ~/terraform/ansible/
- \$ sudo cloud-init status --wait
- \$ make ansible-init
- \$ make ansible-gfarm-install
- \$ make ansible-gfarm-setup

Gfarmのクラウドデプロイ クライアントの利用 (1) OCI内クライアント

- (tfコンテナ) \$./ocissh gfclient01

- (OCI側 gfclient01)

```
[gfarmsys@gfclient01 ~]$ gfmddhost -l
```

```
+ master - m siteA gfmd1.example.org 601
```

```
+ slave sync c siteA gfmd2.example.org 601
```

```
+ slave async c siteB gfmd3.example.org 601
```

```
[gfarmsys@gfclient01 ~]$ gfhost -lv
```

```
0.01/0.18/0.39 T aarch64 1 gfsd01.example.org 600 0(10.0.1.120)
```

```
0.01/0.30/0.57 T aarch64 1 gfsd02.example.org 600 0(10.0.1.146)
```

```
0.01/0.29/0.59 T aarch64 1 gfsd03.example.org 600 0(10.0.1.202)
```

```
0.03/0.36/0.63 T aarch64 1 gfsd04.example.org 600 0(10.0.1.43)
```

Gfarmのクラウドデプロイ クライアントの利用 (2) OCI外クライアント

- (tfコンテナ)
- \$ cd ~/terraform
- \$ make gfarm-install
- \$ make gfarm-config-fetch
- \$ make update-etchosts-for-gfarm
- \$ grid-proxy-init
- \$ gfhost -lv

Gfarmのクラウドデプロイ環境の破棄

- (tfコンテナ) \$ make terraform-destroy
 - yes を入力
 - terraform destroyを1回実行しただけでは消えないのでMakefile側で2回実行している
 - 理由は、DNS情報を消してからでないだとインスタンスを削除できないため
 - 簡単に破棄できるので注意
- tfコンテナの破棄
 - 手順はREADME.md参照
 - 設定ファイルや鍵ファイルは残るので、再利用可能

Gfarmのクラウドデプロイ その他

- Gfarmサーバホスト名解決
 - Gfarmサーバ同士も、ホスト名で通信する必要がある
 - OCI側のインスタンス同士の通信は、OCIのDNSプライベートゾーンで名前解決し、プライベートアドレスを使う
 - Public IP同士でも通信できるが、イングレスルールで通信制限するとしたら設定複雑
 - Terraformでプライベートゾーンを設定するようにした
 - OCI外のクライアント側では、terraform outputの情報からグローバルアドレスを取得し、/etc/hostsを生成
 - DNSに登録しても利用できる想定

GfarmのLustre HSM連携

Lustre HSMとは

- Lustreのファイルを、別のストレージにアーカイブする仕組み
 - テープ
 - S3
 - POSIXファイルシステム
- 「Gfarmシンポジウム2023」でも紹介された
 - <http://oss-tsukuba.org/event/gf2023>
 - 「LustreからGfarmへの透過的なデータアーカイブ」
 - 株式会社データダイレクト・ネットワークス・ジャパン 井原 修一 様
 - S3のCopyツールを使ってGfarmをアーカイブにする方法を紹介いただいた
 - GfarmのS3中継機能 (gfarm-s3-minio) を使って連携

Lustre HSM アーカイブの状態

状態	Lustre	アーカイブ	操作
Lustre側のみデータが存在	存在	無し	<ul style="list-style-type: none">• Lustre側にファイルを作成直後• <code>sudo lfs hsm_remove</code> ファイル名
Lustre側とアーカイブ側にデータが存在	存在	存在	<ul style="list-style-type: none">• <code>sudo lfs hsm_archive</code> ファイル名• リリースされた状態でファイルを読んだ場合• または、<code>lfs hsm_reload</code> ファイル名
Lustre側とアーカイブ側にデータが存在するが、同期されていない(dirty)	存在	存在(不一致)	<ul style="list-style-type: none">• アーカイブされた状態でファイル更新
アーカイブ側のみデータが存在 (Lustre側にはファイルメタ情報のみ)	無し	存在	<ul style="list-style-type: none">• <code>sudo lfs hsm_release</code> ファイル名

Lustre HSM

アーカイブの状態: 実行例(1)

```
$ lfs hsm_state file
```

```
file: (0x00000000)
```

```
$ sudo lfs hsm_archive file
```

```
$ lfs hsm_state file
```

```
file: (0x00000009) exists archived, archive_id:1
```

```
$ lfs hsm_release file
```

```
$ lfs hsm_state file
```

```
file: (0x0000000d) released exists archived, archive_id:1
```

Lustre HSM

アーカイブの状態: 実行例(2)

```
$ lfs hsm_restore file
```

```
$ lfs hsm_state file
```

```
file: (0x00000009) exists archived, archive_id:1
```

```
$ lfs hsm_release file
```

```
$ cat file
```

```
ABCDEFGFG
```

```
$ lfs hsm_state file
```

```
file: (0x00000009) exists archived, archive_id:1
```

```
$ sudo lfs hsm_remove file
```

```
$ lfs hsm_state file
```

```
file: (0x00000000), archive_id:1
```

GfarmのLustre HSM連携 概要

- LustreのHSMアーカイブ側としてGfarmを活用
- Lustre HSM for gfarm2fsを使用して連携
 - <https://github.com/oss-tsukuba/lustre-release/tree/hsm-posix-for-gfarm2fs/hsm-posix>
 - サンプル実装としてLustreに同梱されている POSIX Copyツール (lhmstool_posixコマンド) を改造し、gfarm2fsと組み合わせて使用
- お試し環境構築スクリプト
 - <https://github.com/oss-tsukuba/incus-auto/tree/main/example/gfarm>

GfarmのLustre HSM連携 利用想定

- Lustre領域の容量節約
 - hsm_releaseすれば、ファイル内容はLustre側から削除される
- Lustreに作成したファイルをGfarmからでもファイル読み取り可能に
 - Gfarm側はアーカイブ専用ディレクトリを使用
 - 読み取り専用として使う必要あり (Gfarm -> Lustre に反映できない)
 - Gfarm側のアーカイブ用ファイルを直接更新してはならない
 - (Gfarm側で書き込み権限があるユーザは、ファイルを更新できてしまうので注意)
- できないこと
 - 一般ユーザはLustre HSMのアーカイブ状態を操作変更できない
 - Gfarmに元からあるファイルツリーをそのままLustre側に同期することはできない
 - GfarmからLustreにインポートするコマンドがあるが、期待動作ではない
 - インポートしたファイルをLustre側で更新した場合、Gfarmのインポート元ファイルは更新されない
 - Gfarm側のアーカイブ専用ディレクトリ以下のみが更新される
 - Gfarm側とLustre側ともに読み取り専用としては使ってもよさそう

Lustre HSM for gfarm2fs

仕組み概要

- lhmstool_posixコマンドをベースに改造した
 - Lustreに同梱されているコマンド、Lustre HSMのサンプル実装
 - 大きな変更はしていない
 - gfarm2fsと組み合わせた場合に、Gfarm側で未対応のLustre用拡張属性の操作がエラーになるので変換して保存できるように変更
 - Lustre全体をビルドせずとも、単独でビルドできるようにした
 - gfarm2fsとlhmstool_posixを同時に実行するスクリプト例を用意
 - systemdで起動する手順を用意
- HSMアーカイブ用ディレクトリをGfarmに作成しておく
- gfarm2fsの-o allow_rootオプションを指定してマウント
- ローカルのroot権限でlhmstool_posixを実行

Lustre HSM for gfarm2fs

注意点 (1)

- 自動でアーカイブなどをするなら、別途仕掛けが必要
 - findコマンドによる条件と、アーカイブコマンド組み合わせを定期実行
 - Robinhood Policy Engine (詳細未確認)
 - <https://github.com/cea-hpc/robinhood/wiki>
 - https://robinhood.sourceforge.net/files/LustreHSM_PolicyEngine_v212_admin_guide.pdf
- Gfarm側に所有ユーザ・グループや、ファイルモードを維持する方法
 - Gfarm2fsマウントする際にgfarmrootグループ権限が必要
 - ローカル(Lustre)のユーザ名・グループ名空間と同じユーザ名・グループ名をGfarm側にも登録しておく必要あり
 - とはいえ、Gfarm側のアーカイブ用ファイルを直接更新してはならない
 - Lustre側のファイル属性を変更した場合、Gfarm側に再同期するには、いったんリリースしてからアーカイブする必要あり
 - 維持不要の場合は、lsmstool_posixの--no-attrオプションを使えばエラーにならない

Lustre HSM for gfarm2fs

注意点 (2)

- Gfarm側のLustre HSM専用ディレクトリ以下にアーカイブされる
 - 開始前にGfarm側にディレクトリ作成と、開始時に指定する
 - Gfarm側に作られるファイル実体はLustreのファイルのID由来の名前となり、それに対して実ファイル名のシンボリックリンクが作られる
 - 「Gfarmアーカイブ先/shadow/…」にシンボリックリンクが作られる
- lhmstool_posixを重複実行するとファイルが破損する場合がある
 - 動作確認中に気づいた
 - 起動スクリプト例では、残存プロセスをkillしてから開始するようにした
 - Lustre HSM for gfarm2fsをsystemdで実行する手順
 - <https://github.com/oss-tsukuba/lustre-release/blob/hsm-posix-for-gfarm2fs/hsm-posix/README-gfarm2fs.md>

Lustre HSM for gfarm2fs アーカイブされたGfarm側ファイルツリー

```
[gfarmsys@lclient1 tmp]$ find /mnt/gfarm-lustre-hsm/  
/mnt/gfarm-lustre-hsm/  
/mnt/gfarm-lustre-hsm/0002  
/mnt/gfarm-lustre-hsm/0002/0000  
/mnt/gfarm-lustre-hsm/0002/0000/0bd1  
/mnt/gfarm-lustre-hsm/0002/0000/0bd1/0000  
/mnt/gfarm-lustre-hsm/0002/0000/0bd1/0000/0002  
/mnt/gfarm-lustre-hsm/0002/0000/0bd1/0000/0002/0000  
/mnt/gfarm-lustre-hsm/0002/0000/0bd1/0000/0002/0000/0x200000bd1:0x2:0x0 #ファイル実体  
/mnt/gfarm-lustre-hsm/0002/0000/0bd1/0000/0002/0000/0x200000bd1:0x2:0x0.lov #管理ファイル  
/mnt/gfarm-lustre-hsm/shadow  
/mnt/gfarm-lustre-hsm/shadow/tmp  
/mnt/gfarm-lustre-hsm/shadow/tmp/testfile #シンボリックリンク
```

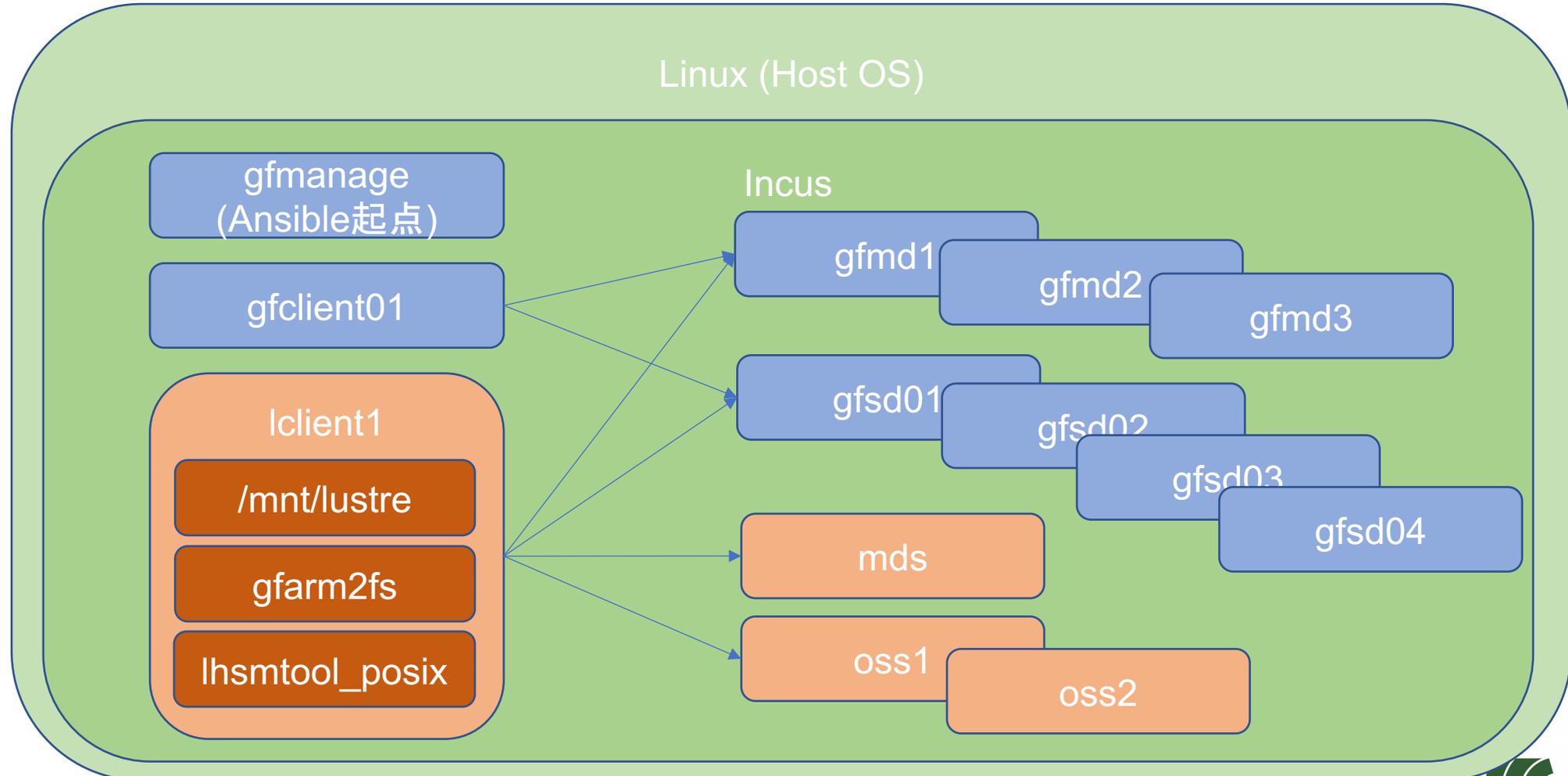
GfarmのLustre HSM連携 環境構築するとしたら・・・

- 通常のGfarm, Lustre構築手順に加えて、
 - Lustre HSM for gfarm2fsをビルド
 - gfarm2fs起動
 - Lustre HSM for gfarm2fsのlhsmtool_posixコマンド起動
- 複数ホストマシン(VMなど)の準備が面倒
 - OSインストール、ネットワークの準備、ホスト名の設定
- Gfarmの環境を複数ホストに構築するのが面倒
- Lustreの環境を複数ホストに構築するのが面倒
- 少し試す目的・開発環境とするには、自動で構築したい

GfarmのLustre HSM連携 お試し環境自動構築

- <https://github.com/oss-tsukuba/incus-auto/tree/main/example/gfarm>
- 1台のLinuxマシン上に、Gfarm+Lustre環境を自動構築
- Gfarmを動かすためにIncusコンテナを使用
- Lustreを動かすためにIncus VMを使用
 - IncusコンテナではLustreが動作しないため
- Incus VMを使うには/dev/kvmが必要
 - Hyper-VやVirtualBoxなどで動かす場合、Nested Virtualizationの有効化が必要
 - WindowsのWSL2の場合、Windows11かつnestedVirtualization=true設定に加えて、WSL2カーネルのビルドが必要
 - (将来ビルド不要になることを期待)
 - macOSの場合、VM(UTM,multipassなど) の中ではVMが動かない
 - (Appleシリコンの場合だけ?)
- incus 6.0.0 (Ubuntu 24.04リポジトリ, 2024/9/1現在) の場合、環境構築できない
 - VMからホストOSのディレクトリをマウントできない
 - Incus本件のリポジトリから取得したバージョン6.0.1や6.4であれば動作した

GfarmのLustre HSM連携 お試し環境 構成図



GfarmのLustre HSM連携 お試し環境 構成

- ホストOS: Linuxマシン
- Incusインスタンス群
 - 管理ホスト (コンテナ)
 - Gfarmクライアント x 1 (コンテナ)
 - Gfarm gfmd x 3 (コンテナ)
 - Gfarm gfsd x 4 (コンテナ)
 - Lustre MGS,MDS x 1 (VM)
 - Lustre OSS x 2 (VM)
 - Lustreクライアント x 1 (VM) (Gfarmにもアクセス可)
- YAMLファイルを書き換えれば、ホスト数や各種サイズなど変更可能

GfarmのLustre HSM連携 お試し環境 注意点 (1)

- メモリ・ディスク使用量
 - ディスク: 約100GB, SSD
 - メモリ: 約10GB
 - ホスト側RAM 16GB, Windows 11 pro Hyper-V の環境で構築できた
 - N100系(N95)CPUのミニPC、RAM16GBの環境で構築できた
- 構築に時間がかかる
 - 30分～1時間以上 (PC性能、ネットワーク次第)
 - Incusの仕組みで共通なシステム一式をなるべく複製している
 - (とはいえコピーにも時間かかる)
 - VMの中でVM動かす場合、環境によってはとても遅い
 - LustreのRPM取得に失敗することがあるので、取得のリトライと再利用可能にした

GfarmのLustre HSM連携 お試し環境 注意点 (2)

- 1台のLinuxマシン内に構築するので、実用向けではない
- RockyLinux 8が更新されると構築できなくなる
 - スクリプト中のLustreバージョン番号変更が必要
 - <https://github.com/oss-tsukuba/incus-auto/blob/main/example/gfarm/SCRIPT/install-lustre-common.sh>
 - 新しいLustreパッケージのバージョンを指定する

GfarmのLustre HSM連携 お試し環境構築スクリプトの仕組み

- Gfarm環境とLustre環境をIncus上に構築
 - incus-autoを使用してIncusのコンテナ・VM群を構築
 - Gfarmのインストール
 - Lustreのインストール
 - (Lustre無しでGfarmのみの環境も構築可能)
- Ansibleを使用してGfarmの環境構築

GfarmのLustre HSM連携 お試し環境構築手順 (一つずつ実行する場合)

- \$ cd incus-auto/example/gfarm/
- \$ make git-clone
- \$ make init
- \$ make build-gfarm
- \$ make launch-gfarm
- \$ make build-lustre
- \$ make launch-lustre
- \$ make setup-gfarm-all

GfarmのLustre HSM連携 お試し環境 構築手順 (一度に自動構築)

- \$ make REBUILD-ALL
 - Gfarm + Lustre
- \$ make REBUILD-PARA-ALL
 - 構築を並列に実行
- \$ make REBUILD-GFARM
 - Gfarmのみ

GfarmのLustre HSM連携 お試し環境 Lustre HSM for gfarm2fs起動・利用

- \$ make start-lustre-hsm
 - Ctrl-Cするまで停止しない
- (別の端末を使用)
- \$ make lclient
- (Lustreクライアントホストのシェルを利用)
- \$ cd /mnt/lustre
- \$ sudo mkdir -m 1777 tmp
- \$ cd tmp
- (以降、Lustreの操作)

まとめ

- Gfarmのクラウドデプロイ
 - Oracle Cloud InfrastructureにGfarmをデプロイする仕組み紹介
 - [incus-auto/example/gfarm-terraform-oci](#)
- GfarmのLustre HSM連携
 - Lustre HSM (Hierarchical Storage Management) 概要
 - Lustre HSMとしてGfarmにアーカイブする方法紹介
 - お試し環境自動構築手順
 - [incus-auto/example/gfarm](#)
- そのまま活用できる仕組みではないが、ご参考になれば

デモ用

incus-auto/example/gfarm-terraform-oci

- `$ make shell`
- `gfarmsys@tf:~$ cd terraform`
- `gfarmsys@tf:~/terraform$ terraform apply`
- `gfarmsys@tf:~/terraform$ make update-ssh_known_hosts`
- `gfarmsys@tf:~/terraform$ make update-ssh_config`
- `gfarmsys@tf:~/terraform$ make update-ansible-inventory`
- `gfarmsys@tf:~/terraform$ make send-files`
- `gfarmsys@tf:~/terraform$./ocissh gfmanage`
- `gfarmsys@gfmanage:~$ cd terraform/ansible/`
- `gfarmsys@gfmanage:~/terraform/ansible$ make ansible-init`
Do you want to continue? [Yes/No]: y
- `gfarmsys@gfmanage:~/terraform/ansible$ time make ansible-gfarm-install`
- `gfarmsys@gfmanage:~/terraform/ansible$ time make ansible-gfarm-setup`
- `gfarmsys@gfmanage:~/terraform/ansible$ logout`
- `gfarmsys@tf:~/terraform$./ocissh gfclient01`
- `[gfarmsys@gfclient01 ~]$ gfdf -h`

デモ用

incus-auto/example/gfarm

- (環境構築済み)
- ~/incus-auto/example/gfarm\$ make start-lustre-hsm
- (別の端末)
- ~/incus-auto/example/gfarm\$ make lclient
- [gfarmsys@lclient1 tmp]\$ cd /mnt/lustre/tmp/
- [gfarmsys@lclient1 tmp]\$ date > testfile
- [gfarmsys@lclient1 tmp]\$ sudo lfs hsm_state testfile
- [gfarmsys@lclient1 tmp]\$ sudo lfs hsm_archive testfile
- [gfarmsys@lclient1 tmp]\$ sudo lfs hsm_state testfile
- [gfarmsys@lclient1 tmp]\$ ls -l /mnt/gfarm-lustre-hsm/shadow/tmp/testfile
- [gfarmsys@lclient1 tmp]\$ sudo lfs hsm_release testfile
- [gfarmsys@lclient1 tmp]\$ ls -l /mnt/gfarm-lustre-hsm/shadow/tmp/testfile
- [gfarmsys@lclient1 tmp]\$ sudo lfs hsm_state testfile
- [gfarmsys@lclient1 tmp]\$ cat testfile
- [gfarmsys@lclient1 tmp]\$ sudo lfs hsm_state testfile