

# OpenPubKey によるイメージの署名と検証

NTT ソフトウェアイノベーションセンタ  
須田 瑛大

- Moby (Docker Engineのupstream)、BuildKit、containerd、runcなどのOSSのコミッタ
- 今年のDockerConでは “Reproducible builds with BuildKit for software supply chain security” に関して登壇
  - <https://medium.com/nttlabs/dockercon-2023-reproducible-builds-with-buildkit-for-software-supply-chain-security-0e5aedd1aaa7>
- 今日は自身の取り組みについてではなく、Docker社・BastionZero社によるOpenPubKeyへの取り組みについて紹介
- Reproducible builds にも OpenPubKey を応用していきたいと思っている

# DockerCon での OpenPubKey 関連セッション

NTT 

- **2023/10/4: Demystify Secure Supply Chain Metadata**
  - Christian Dupuis, Sr Principal Engineer, Docker
- **2023/10/5: Building the Software Supply Chain on Docker Official Images**
  - Ethan Heilman, CTO & Co-Founder, BastionZero
  - James Carnegie, Computer Programmer, Docker
- <https://www.dockercon.com/> からまだ視聴可能

# Linux Foundation プロジェクト化のアナウンス NTT

The screenshot shows a web browser window displaying a news article from the Linux Foundation's website. The article is titled "Linux Foundation, BastionZero and Docker Announce the Launch of the OpenPubkey Project". It was written by Jason Perlow on October 4, 2023, and is estimated to be a 2-minute read. The article discusses the launch of the OpenPubkey project, which aims to advance the UN's sustainability goals through open source technology. The Linux Foundation logo is visible at the top of the page, along with a 'JOIN' button. The background features a faint globe graphic.

## OpenPubkey

# OpenPubKey

- OpenID Connect (OIDC) を応用し、公開鍵基盤として使えるようにしたもの  
*"In essence, OpenPubkey is a protocol for getting OpenID Providers (OPs) to bind identities to public keys."* (<https://github.com/openpubkey/openpubkey>)
- 例えば、GitHub Actions 上でDockerイメージをビルドする際に、GitHubリポジトリのorganization IDを含む署名を付加できる
- GitHub Actions の Secrets に秘密鍵をアップロードしたりしなくてよい
- GitHub Actions 以外の OpenID プロバイダへの対応も容易。  
プロバイダ側での対応作業は(原則的に)不要。
- Sigstore の Cosign (keyless) に似ているが、よりシンプルな構成

# DockerConでのOpenPubKeyのデモ



- GitHub ActionsのYAML:

<https://github.com/openpubkey/demo>

- BuildKitへのパッチ (未マージ):

<https://github.com/openpubkey/buildkit/tree/opk-signing>

- docker verify コマンド:

<https://github.com/openpubkey/verify-docker-cli-plugin>

# GitHub Actions の YAML



```
- name: Setup Docker buildx
  uses: docker/setup-buildx-action@v3.0.0
  with:
    driver-opts: |
      image=openpubkey/buildkit:opk-signing
      env.ACTIONS_ID_TOKEN_REQUEST_URL=${{ env.ACTIONS_ID_TOKEN_REQUEST_URL }}
      env.ACTIONS_ID_TOKEN_REQUEST_TOKEN=${{ env.ACTIONS_ID_TOKEN_REQUEST_TOKEN }}

- name: Build and push Docker image on push
  id: build-and-push
  uses: docker/build-push-action@v4.0.0
  with:
    context: .
    push: true
...
    sbom: true
    provenance: true
...
```

# GitHub Actions の YAML

```
- name: Setup Docker
  uses: docker/https://github.com/openpubkey/buildkit
  with:
    driver-opts: |
      image=openpubkey/buildkit:opk-signing
      env.ACTIONS_ID_TOKEN_REQUEST_URL=${{ env.ACTIONS_ID_TOKEN_REQUEST_URL }}
      env.ACTIONS_ID_TOKEN_REQUEST_TOKEN=${{ env.ACTIONS_ID_TOKEN_REQUEST_TOKEN }}
```

OpenPubKeyに対応した fork を指定する  
<https://github.com/openpubkey/buildkit>

GitHub により自動的に設定される  
OIDC関連の環境変数を  
BuildKit デーモン に伝播させる

```
- name: Build and push Docker image on push
  id: build-and-push
  uses: docker/build-push-action@v4.0.0
  with:
    context: .
    push: true
...
    sbom: true
    provenance: true
```

SBOMおよびSLSA Provenance (ビルド情報) を  
イメージに付加する。  
併せて、SBOMやSLSA Provenance に対する署名  
(i.e., イメージに対する署名) も付加する。

# docker verify コマンド

- Docker CLI プラグインとして提供

<https://github.com/openpubkey/verify-docker-cli-plugin>

```
$ gh api /orgs/openpubkey | jq .id  
145685596  
  
$ docker verify openpubkey/demo@sha256:6acf54fca0e7f9e9ef3d23c9b9ee306ce3c56b8315aa756643e4df25032bcdbe \  
--repo-owner-id 145685596  
...  
Verified all required attestations are present
```

- Docker Hub 上のイメージ “[openpubkey/demo@sha256...](#)” が、GitHub organization “[openpubkey](#)” の GitHub Actions でビルドされたことを確認できた
- ただし、GitHub Actions のキーセット (<https://token.actions.githubusercontent.com/.well-known/jwks>) を信頼できる前提

# docker verify コマンド (出力全体)

```
$ docker verify openpubkey/demo@sha256:6acf54fca0e7f9e9ef3d23c9b9ee306ce3c56b8315aa756643e4df25032bcdbe \
--repo-owner-id 145685596
Verifying 2 attestations for
pkg:docker/openpubkey/demo?digest=sha256:6acf54fca0e7f9e9ef3d23c9b9ee306ce3c56b8315aa756643e4df25032bcdbe&platform
=linux/amd64

Verifying https://spdx.dev/Document attestation
✓ Verified attestation refers to digest sha256:6fb78f2482a527575558d85147aee5f5f9f5288133bb0af316f91ed525a7246
✓ Verified OIDC token was signed by https://token.actions.githubusercontent.com
✓ Verified attestation digest was signed on a Github Actions run:
https://github.com/openpubkey/demo/actions/runs/6628798402
✓ Verified repository owner 145685596 (openpubkey)

Verifying https://slsa.dev/provenance/v0.2 attestation
✓ Verified attestation refers to digest sha256:6fb78f2482a527575558d85147aee5f5f9f5288133bb0af316f91ed525a7246
✓ Verified OIDC token was signed by https://token.actions.githubusercontent.com
✓ Verified attestation digest was signed on a Github Actions run:
https://github.com/openpubkey/demo/actions/runs/6628798402
✓ Verified repository owner 145685596 (openpubkey)
✓ Verified signed git sha provenance 6b29027d7ddae1c2dcdfb88ab402473af5d20c8
✓ Verified signed git repo provenance openpubkey/demo

Verified all required attestations are present
```

# フロー (1/2)

BuildKit  
(OpenPubKey対応版)

- ・イメージをビルドし、provenanceを生成
- ・一回限りの鍵ペアを生成
- ・Client-Instance Claim (CIC) を生成  
{'rz': random(),  
 'upk': publicKey,  
 'alg': 'EC256',  
 'att': provenenceHash}
- ・CIC のハッシュを "audience" としてID Tokenの発行を要求

```
$ACTIONS_ID_TOKEN_REQUEST_TOKEN  
$ACTIONS_ID_TOKEN_REQUEST_URL
```

GitHub Actions

- ・ID Token を発行するための Bearer Token を発行

```
HTTP GET  
$ACTIONS_ID_TOKEN_REQUEST_URL  
&audience=$AUDIENCE  
“Authorization: bearer  
$ACTIONS_ID_TOKEN_REQUEST_TOKEN”
```

audience (GHAのデフォルトではリポジトリのURL) には任意の文字列をいれてよいのでOIDCのAPIを変更せずにCICのハッシュを送ることができ

# フロー (2/2)

(前掲)

CIC: Client-Instance Claim: {'rz': random(), 'upk': publicKey, 'alg': 'EC256', 'att': provenanceHash}  
audience: ここでは CIC のハッシュ



## BuildKit (OpenPubKey 対応版)

ID Token が鍵ペアに紐づく

## GitHub Actions

- 手持ちのCICと、GHAから受け取ったID Tokenとを組み合わせ、署名し、PK Token を構成する。これをイメージに付加する。
- Guillou-Quisquater (GQ) 署名により、元のID Tokenの値は隠蔽される（ペイロードの部分は公開される）

ID Token: "SUBSTR0.SUBSTR1.SUBSTR2"  

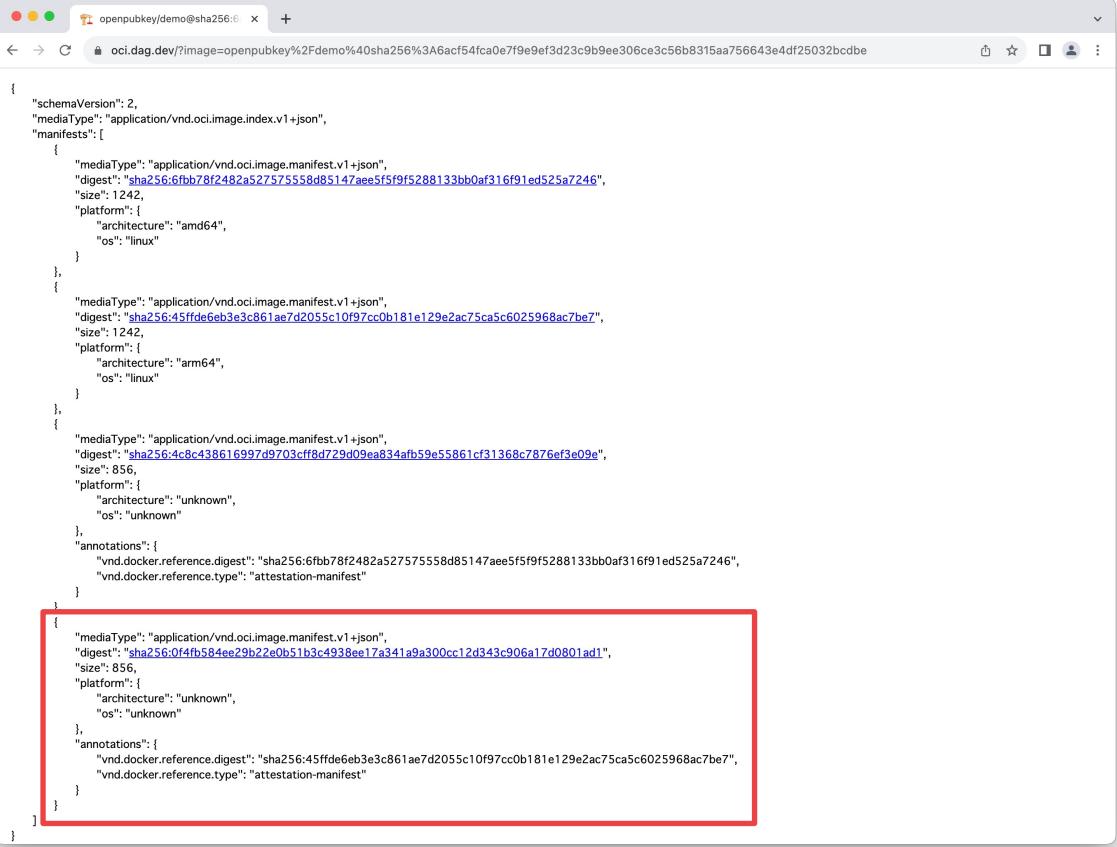
- SUBSTR0: ヘッダの base64
- SUBSTR1: ペイロードの base64 ("aud"などを含む)
- SUBSTR2: "FOO.BAR" に対する署名

- 受け取った Bearer Token が合っていれば、ID Token を発行する
- ID Token の主なペイロード:
  - "aud" (audience)
  - "repository\_owner\_id" (GitHub の org の ID)
  - "run\_id" (GHA の job の実行 ID)
- ID Token は、GHA 自身が持っている鍵で署名される

GHA の場合はペイロードが公開されて問題ないが、他の OpenID プロバイダでは検討が必要そう

※ GQ 署名は workload identity では必須、user identity では任意

# 実際のデータ



```
{ "schemaVersion": 2,
  "mediaType": "application/vnd.oci.image.index.v1+json",
  "manifests": [
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "digest": "sha256:6fbb78f2482a527575558d85147aee5f5f9f5288133bb0af316f91ed525a7246",
      "size": 1242,
      "platform": {
        "architecture": "amd64",
        "os": "linux"
      }
    },
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "digest": "sha256:45ffde6eb3e3c861ae7d2055c10f97cc0b181e129e2ac75ca5c6025968ac7be7",
      "size": 1242,
      "platform": {
        "architecture": "arm64",
        "os": "linux"
      }
    },
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "digest": "sha256:4c8c438616997d9703cff8d729d09ea834afb59e55861cf31368c7876ef3e09e",
      "size": 856,
      "platform": {
        "architecture": "unknown",
        "os": "unknown"
      },
      "annotations": {
        "vnd.docker.reference.digest": "sha256:6fbb78f2482a527575558d85147aee5f5f9f5288133bb0af316f91ed525a7246",
        "vnd.docker.reference.type": "attestation-manifest"
      }
    },
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "digest": "sha256:0f4fb584ee29b22e0b51b3c4938ee17a341a9a300cc12d343c906a17d0801ad1",
      "size": 856,
      "platform": {
        "architecture": "unknown",
        "os": "unknown"
      },
      "annotations": {
        "vnd.docker.reference.digest": "sha256:45ffde6eb3e3c861ae7d2055c10f97cc0b181e129e2ac75ca5c6025968ac7be7",
        "vnd.docker.reference.type": "attestation-manifest"
      }
    }
  ]
}
```

<https://oci.daq.dev/?image=openpubkey%2Fdemo%40sha256%3A6acf54fca0e7f9e9ef3d23c9b9ee306ce3c56b8315aa756643e4df25032bcdbe>

# 実際のデータ

```
{  
    "mediaType": "application/vnd.oci.image.manifest.v1+json",  
    "digest": "sha256:0f4fb584ee29b22e0b51b3c4938ee17a341a9a300cc12d343c906a17d0801ad1",  
    "size": 856,  
    "platform": {  
        "architecture": "unknown",  
        "os": "unknown"  
    },  
    "annotations": {  
        "vnd.docker.reference.digest": "sha256:45ffde6eb3e3c861ae7d2055c10f97cc0b181e129e2ac75ca5c6025968ac7be7",  
        "vnd.docker.reference.type": "attestation-manifest"  
    }  
}  
  
vnd.docker.reference.type : attestation-manifest  
}  
  
{  
    "mediaType": "application/vnd.oci.image.manifest.v1+json",  
    "digest": "sha256:0f4fb584ee29b22e0b51b3c4938ee17a341a9a300cc12d343c906a17d0801ad1",  
    "size": 856,  
    "platform": {  
        "architecture": "unknown",  
        "os": "unknown"  
    },  
    "annotations": {  
        "vnd.docker.reference.digest": "sha256:45ffde6eb3e3c861ae7d2055c10f97cc0b181e129e2ac75ca5c6025968ac7be7",  
        "vnd.docker.reference.type": "attestation-manifest"  
    }  
}
```

# 実際のデータ



https://oci.dag.dev/blob/openpubkey/demo@sha256:f5daba3a4bcd7f76563a5ff36f6dab8f6e5e705d7b23e541a5913946a3a08bde/?mt=application%2Fvnd.in-toto.provenance%2Bdsse&size=16563

openpubkey/demo@sha256:f5daba3a4bcd7f76563a5ff36f6dab8f6e5e705d7b23e541a5913946a3a08bde

Registry Explorer

Content-Type: application/vnd.in-toto.provenance+dsse

Docker-Content-Digest: sha256:f5daba3a4bcd7f76563a5ff36f6dab8f6e5e705d7b23e541a5913946a3a08bde

Content-Length: 16563

\$ crane blob openpubkey/demo@sha256:f5daba3a4bcd7f76563a5ff36f6dab8f6e5e705d7b23e541a5913946a3a08bde

```
{"payloadType": "application/vnd.in-toto+json", "payload": "eyJfdflHwzSl6ImhoDbBzI0vaW4tDg90by5phy9TdfGZPwYjBnQjdRhwcz0l3Nsc2EzGV2L3byb32JbmFuPy2UvdgAuMlsInN1YmpY3Qj0l7t5hbWUj0iJwa2cZGzq2aVby2L9wZW5wdVbXZkvvZGvBb0b2tYWWl3bsXrbm3JtRwxbnvri4JTGYJXNjQlCjkaWdlc3QOnisc2hmmHU2j0lDm7mRnNmVbM2UzYz2MFN2QyMDU1YzEwZj3Y2MwJy4MWMUxMjlMmPfJn2VjYTJNjAyNT2k0GFJN2JNjY9Fv0snb2yZwRpY2F0ZSi6geJldWzLsGZV9ypl7imlkjloiaHRoCHM6Ly9naXroDwUy29tL29wZWSwdWJrXzvZGVb9yH9Y3Rpz2S3J1bnmVnjYyODc5ODQyMj9LCLjdWzLsFR5cGU0JodJhRwczoVl21tVnlwvcmnqzWN0Lm9yZy9ldWzLsGtDpzbM5lsm1h1gdHgYaWfFscy6W3sdicPjliojcGtnOmRvYtcl09kb2nZxlVnRpbGraxQtc3lmdc12YFubmVvQjHNOYjwzS0xiwzGlnZxN0jpj7tNoYt11Ni6mVm2cYnNND2E2zmJm21mWuLyTgZqMTQ4MWMyYm1tY2QvNje2NjyjUyTAQY2Q5MDjYjYWewZDExMTWOGQ20DMdf0xsey1cmkiJluwz6cZGz9j2avYl25vZGVAYVwxaV5IP3BsYRmb3JtPwxbpnV4JtJGJYXjtJnQlCjkaWdlc3Q0nsic2hmmU2j0lJm2lziJkONWQ1NmyWU3Y3NnYtTuNxMyTdhMdcpwDOAaVnVWQ1ZTVmYz4KMTU0M2NjYczNjWjM2JMDfYj19fVosilmudm9jYXRpzb24iOnsi29uZmlnu291cnNljp7tVnUdJJSUG9pbnQj0lJeB2NrzXjmaWxlN0snRhcmtzXrXcmIOnsiJnVbRlmQj0lkb2NzXjmaWxlN0tVnliwYJnct6eyjsYtWjlbDpvcmcu3BlbmNvnRhaW5cnuW1h22Uu3JYJXRZC16jJwMjMtTAitMjRUMTU6MDY6NDEuMDQ4WilsImxhYrnVs0m9jyZvCgVu729udGfbpmVcy5pbWnfZS5kZxNjmmlwdGvbl6llsismxhYmvs0m9yZvCgVu729udGfbpmVcy5pbWnfZSSsaWnlbnRnCly6kfWvWn0zSoyAilCjSjyWjlbDpvcmcu38lbnmVbnRhaW5cnuW1hZ2UucmV2XNbpz24i0l2Yj5MD13Zdk2ZGmWYzGK9Y2ZlDhyQjwmQ3M2FmNWQyMGM4lwibGfZVw6b3JmLn9wzW5jb2505WluXjLmItVwddlnNvdxJzS6lmlm0hdBz0l8vZl0ahVlNmVs9vcVuchVia2V5L2RlbW8lCjss2RlbW8lCj5yJlbDpvcmcu38lbnmVbnRhaW5cnuW1h22UudGlobGUj0ljkZw1iwlbGfzWw6b3JnLn9wzW5jb250YluXzJzLmItVwddlnNvdybC6lmh0dHbz0l8vZl0ahVlNmVs9vcVuchVia2V5L2RlbW8lCjssYwJlbDpvcmc9ybsGIGlmxbp9ybs9u2BzR0OmleS59p1wXslj6lGmxpbn4InleS5qc1wXslj0jEsIm9t1dHRIc3RQYXRoUj5x7mIlkjicj3RlkMTEwMj4ANdij2jYyZrRzT
```

デコードしないと読めない

<https://onlinejsontoolt.com/convert-base64-to-json>

```
application/vnd.in-toto.provenance+dsse
```

```
{  
  "payloadType": "application/vnd.in-toto+json",  
  "payload": ...,  
  "signatures": [{"keyid": "OPK", "sig": ...}]  
}
```

```
.signatures[0].sig  
  
{  
  "payload": ...,  
  "signatures": [  
    {"protected": ..., "header": {"sig_type": "cic"}, "signature": ...},  
    {"protected": ..., "header": {"sig_type": "oidc_gq"}, "signature": ...}  
  ]  
}
```

```
.signatures[0].protected  
  
{  
  "alg": "ES256",  
  "att": ...  
  "rz": ...  
  "upk": {"alg": "ES256", "crv": "P-256",  
          "kty": "EC", "x": ..., "y": ... }  
}
```

.payload

(SLSA Provenance)

.payload

```
{  
  "jti": "66e345cc-0806-42a0-a346-7b98dc8ab2c0",  
  "sub": "repo:openpubkey/demo:ref:refs/heads/main",  
  "aud": "5km4t6i4IPk-tnv4qlmCcryfY00-4PCNwOmFCJZJrzI",  
  "ref": "refs/heads/main",  
  "sha": "6b29027d7ddae1c2dcdfb88ab402473af5d20c8",  
  "repository": "openpubkey/demo",  
  "repository_owner": "openpubkey",  
  "repository_owner_id": "145685596",  
  "run_id": "6628798402",  
  "repository_id": "698282570",  
  "iss": https://token.actions.githubusercontent.com  
  ...  
}
```

.signatures[1].protected

```
{  
  "typ": "JWT",  
  "alg": "RS256",  
  "x5t": "AB3c0BSoSOiCRXez5P0u2zvPX_0",  
  "kid": "001DDCD014A848E8824577B3E4F3AEDB3BCF5FFD"  
}
```

# まとめ (再掲)

- OpenID Connect (OIDC) を応用し、公開鍵基盤として使えるようにしたもの  
*"In essence, OpenPubkey is a protocol for getting OpenID Providers (OPs) to bind identities to public keys."* (<https://github.com/openpubkey/openpubkey>)
- 例えば、GitHub Actions 上で Dockerイメージをビルドする際に、GitHubリポジトリの organization IDを含む署名を付加できる
- GitHub Actions の Secrets に秘密鍵をアップロードしたりしなくてよい
- GitHub Actions 以外の OpenID プロバイダへの対応も容易。  
プロバイダ側での対応作業は(原則的に)不要。
- Sigstore の Cosign (keyless) に似ているが、よりシンプルな構成

スライド: <https://github.com/AkihiroSuda>  
("Presentation slides" → "2023" → "20231102 ...")