Bridgelet

Getting Started Guide

> **Document Version:** 0.1 (MVP)
> **Last Updated:** January 2025
> **Estimated Time:** 30 minutes
> **Difficulty:** Beginner

1. Welcome to Bridgelet

This guide will help you set up Bridgelet and send your first payment to a recipient without a crypto wallet. By the end, you'll understand how to create ephemeral accounts, send claim links, and manage the payment lifecycle.

1.1 What You'll Need

- Node.js 18+ and npm/yarn installed
- A Stellar testnet account with XLM (we'll help you create one)
- Basic knowledge of JavaScript/TypeScript
- 15-30 minutes of your time

1.2 What You'll Build

A simple payment system that:

1. Creates an ephemeral Stellar account
2. Funds it with test XLM
3. Generates a claim link for a recipient
4. Allows the recipient to claim funds to their wallet

2. Prerequisites Setup

Step 1: Install Node.js

Ensure you have Node.js 18 or higher installed:

```
node --version
```

If not installed, download from nodejs.org

Step 2: Create a Project Directory

```
 mkdir bridgelet-demo
cd bridgelet-demo
npm init -y
```

Step 3: Install Bridgelet SDK

```
 # Install the SDK (testnet version)
npm install @bridgelet/sdk

# Install peer dependencies
npm install @stellar/stellar-sdk dotenv
```

💡 **Tip:** The SDK is currently in development. For MVP testing, clone the repository directly:

```
 git clone https://github.com/bridgelet-org/bridgelet-
cd bridgelet-sdk
```

```
npm install
npm run build
```

3. Configuration

Step 4: Set Up Environment Variables

Create a `.env` file in your project root:

```
 # Stellar Network Configuration
STELLAR_NETWORK=testnet
STELLAR_HORIZON_URL=https://horizon-testnet.stellar.o

# Your organization's Stellar account (we'll create t
ORGANIZATION_SECRET_KEY=your_secret_key_here
ORGANIZATION_PUBLIC_KEY=your_public_key_here

# Bridgelet SDK Configuration
BRIDGELET_API_URL=http://localhost:3000
BRIDGELET_API_KEY=test_key_development
```

⚠️ **Important:** Never commit `.env` to version control. Add it to `.gitignore` immediately.

Step 5: Create a Testnet Account

Create a simple script `setup.js` :

```
 const StellarSdk = require('@stellar/stellar-sdk');

async function createTestnetAccount() {
  // Generate a new keypair
  const pair = StellarSdk.Keypair.random();

  console.log('Public Key:', pair.publicKey());
  console.log('Secret Key:', pair.secret());
```

```
  // Fund the account using Friendbot
  const response = await fetch(
    `https://friendbot.stellar.org?addr=${pair.public
  );

  const result = await response.json();
  console.log('Account funded!', result);

  console.log('\nAdd these to your .env file:');
  console.log(`ORGANIZATION_PUBLIC_KEY=${pair.publicK
  console.log(`ORGANIZATION_SECRET_KEY=${pair.secret(
}

createTestnetAccount();
```

Run it:

```
node setup.js
```

Copy the output keys to your `.env` file.

4. Your First Payment

Step 6: Initialize the SDK

Create `index.js`:

```
 require('dotenv').config();
const { BridgeletSDK } = require('@bridgelet/sdk');

// Initialize the SDK
const bridgelet = new BridgeletSDK({
  network: process.env.STELLAR_NETWORK,
  horizonUrl: process.env.STELLAR_HORIZON_URL,
  organizationSecretKey: process.env.ORGANIZATION_SEC
});

console.log('✅ Bridgelet SDK initialized!');
```

Step 7: Create an Ephemeral Account

Add this function to create your first ephemeral account:

```
async function createPayment() {
 try {
   // Create an ephemeral account with 10 XLM
   const account = await bridgelet.createEphemeralAc
     amount: '10',
     asset: 'XLM',
     recipientEmail: 'recipient@example.com',
     expiresIn: '7d', // Expires in 7 days
     metadata: {
       purpose: 'Test payment',
       orderId: 'ORDER-123'
     }
   });

   console.log('🎉 Ephemeral account created!');
   console.log('Account ID:', account.id);
```

```
    console.log('Public Key:', account.publicKey);
    console.log('Claim URL:', account.claimUrl);
    console.log('Expires:', account.expiresAt);

    return account;
  } catch (error) {
    console.error('❌ Error:', error.message);
  }
}

createPayment();
```

Step 8: Run Your First Payment

```
node index.js
```

You should see output like:

```
✅ Bridgelet SDK initialized!
🎉 Ephemeral account created!
Account ID: abc123...
Public Key: GXXXXXXX...
Claim URL: https://claim.bridgelet.org/claim/abc123
Expires: 2026-01-30T00:00:00Z
```

> 💡 **Success!** You've created your first ephemeral account. The recipient can now use the claim URL to receive their funds.

5. Testing the Claim Flow

Step 9: Simulate a Claim

Add a claim simulation function:

```
async function simulateClaim(accountId) {
  // Create a recipient wallet
  const recipientKeypair = StellarSdk.Keypair.random(

  console.log('Recipient wallet:', recipientKeypair.p

  try {
    // Claim the funds
    const result = await bridgelet.claimFunds({
      accountId: accountId,
      destinationPublicKey: recipientKeypair.publicKe
      verificationCode: 'test-code-123' // In product
    });

    console.log('✅ Funds claimed successfully!');
    console.log('Transaction:', result.transactionId)
    console.log('Amount transferred:', result.amount)

    return result;
  } catch (error) {
    console.error('❌ Claim failed:', error.message);
  }
}

// Run the full flow
async function runDemo() {
  const account = await createPayment();

  if (account) {
    console.log('\nWaiting 5 seconds before claiming.
    await new Promise(resolve => setTimeout(resolve,

    await simulateClaim(account.id);
  }
}
```

```
runDemo();
```

6. Understanding the Flow

6.1 What Just Happened?

1. **Account Creation:** Bridgelet created a temporary Stellar account
2. **Funding:** Your organization's account funded the ephemeral account with 10 XLM
3. **Claim Link Generation:** A unique, secure claim URL was created
4. **Claim Execution:** The recipient connected their wallet and claimed the funds
5. **Auto-Sweep:** Funds automatically transferred to recipient's permanent wallet
6. **Account Closure:** The ephemeral account was closed and removed

6.2 Check the Blockchain

You can verify the transactions on Stellar's testnet explorer:

1. Visit stellar.expert/explorer/testnet
2. Search for the ephemeral account's public key
3. View the creation, funding, and sweep transactions

7. Common Operations

7.1 Check Account Status

```
async function checkStatus(accountId) {
  const status = await bridgelet.getAccountStatus(account

  console.log('Status:', status.state); // 'created', 'fu
  console.log('Balance:', status.balance);
  console.log('Claimed:', status.isClaimed);
  console.log('Expired:', status.isExpired);
}
```

7.2 Handle Expired Accounts

```
async function recoverExpiredFunds(accountId) {
  try {
    const result = await bridgelet.recoverExpiredAccount(

    console.log('✅ Funds recovered to organization walle
    console.log('Amount:', result.amount);
    console.log('Transaction:', result.transactionId);
  } catch (error) {
    console.error('❌ Recovery failed:', error.message);
  }
}
```

7.3 List All Accounts

```
async function listAccounts() {
  const accounts = await bridgelet.listAccounts({
    status: 'pending', // 'pending', 'claimed', 'expired'
    limit: 10,
    offset: 0
  });

  accounts.forEach(account => {
    console.log(`${account.id}: ${account.amount} ${accou
```

```
  });
}
```

8. Next Steps

8.1 Integration Options

Use Case	Recommended Approach
Payroll System	Batch account creation, CSV import support
Aid Distribution	SMS-based claim links, low-bandwidth UI
Rewards/Airdrops	Social verification, anti-fraud checks
Marketplace Payouts	Webhook integration, instant claims

8.2 Additional Resources

- 📘 Architecture Overview - Deep dive into system design
- 🔧 Integration Guide - Production deployment guide
- 🔒 Security Model - Security best practices
- 💡 Use Cases - Real-world examples

8.3 Join the Community

- GitHub: github.com/bridgelet-org
- Discussions: Share your use case and get help
- Issues: Report bugs and request features
- Discord: (Coming soon) Chat with other developers

> 💡 **Ready for production?** Check the Integration Guide for deployment best practices, mainnet setup, and scaling considerations.